MASTER EN SCIENCES INFORMATIQUES

Développement d'un outil de traitement mathématique sur UNIX

Vanden Berghe, M.P.

Award date:
1983

Awarding institution:
Universite de Namur

FACULTES UNIVERSITAIRES N.D. DE LA PAIX

NAMUR

INSTITUT D'INFORMATIQUE

DEVELOPPEMENT D'UN OUTIL DE TRAITEMENT MATHEMATIQUE SUR UNIX

PROMOTEUR :

PH. VAN BASTELAER

M.P. VANDEN BERGHE

ANNEE ACADEMIQUE : 1982.83

Après m'avoir conseillée dans le choix du sujet, Monsieur Ph. Van Bastelaer a bien voulu guider ma démarche tout au long de mes recherches et me conseiller lors de la rédaction de ce travail.

Ses assistants, Mademoiselle B. Scoyer et Monsieur J.-P. Adans, m'y ont aimablement et efficacement aidée.

Je tiens à leur en exprimer ici ma reconnaissance et mes remerciements.

TABLE DES MATIERES

TABLE DES MATIERES

INTRODUCTION

Dans tous les secteurs de l'industrie,du commerce et de l'administration, le bureau reste la plaque tournante de l'information textuelle.Mais c'est le traitement de textes qui rend plus efficace ses differentes activites.

Ce memoire a pour principal objet de traiter des documents contenant des formules mathematiques sur le systeme UNIX des facultes.

Les recents progres spectaculaires realises dans l'equipement des materiels de traitement de textes postulent une etude continue des posssibilites nouvelles ainsi offertes aux utilisateurs.Quels sont les objectifs poursuivis,le champ d'application,la configuration generale et les caracteristiques fonctionnelles de ces nouveaux systemes ? Tels sont les points developpes dans le premier chapitre de ce memoire.

Le deuxieme chapitre analyse le problème de formatage et ses relations avec les autres aspects du traitement de texte.Quelques systemes sont mentionnes et evalues.Ils permettent d'esquisser les perspectives offertes par les systemes futurs.

Sur le systeme UNIX,nous disposons d'un programme de formatage,"NROFF",muni de ses jeux de macros-instructions.Il permet la mise en page de textes mais il est insuffisant si on veut introduire des donnees mathematiques et des tableaux.Voila le problème auquel nous nous attachons dans la suite du travail.

Les difficultes rencontrees lors de l'elaboration des equations mathematiques sont de deux types : d'une part elles necessitent des caracteres,des tailles et des polices divers,d'autre part,elles s'etendent sur deux dimensions.

Etant donne les outils dont on dispose,le chapitre 3 a pour objet d'elaborer un schema d'edition,de formatage et de visualisation qui satisfait aux exigences d'un utilisateur et qui repond aux contraintes techniques posees.
Le chapitre suivant analyse les differentes demarches de realisation, developpe chacune d'elles et presente en detail le processus propose.
Les difficultes et quelques problèmes rencontres en cours d'elaboration sont relates au cinquieme chapitre.

Nous avons relegue en annexes quelques definitions de mots techniques ,signale par le sigle [*] a leur premiere occurrence dans le texte.

Differents tests y sont regroupes.Ils illustrent quelques possibilites d'exploitation du systeme propose.

L'annexe 3 constitue un manuel d'utilisation du systeme destine aux profanes en mathematiques,toutefois inities au maniement de base de l'ordinateur (edition,...).

Les trois annexes suivantes regroupent les commandes specifiques a la Sanders,au VT100 et a Neqn.Elles offrent les details necessaires a une comprehension precise de la realisation de bacK_space,ecran et prefiltre.Leur analyse ainsi que leur implementation cloturent ce travail.

Chapitre 1: INTRODUCTION AU TRAITEMENT DE DOCUMENTS : APERCU DU MARCHE

1. DE LA FRAPPE AUTOMATIQUE AU TRAITEMENT DE TEXTE.

Qu'il s'agisse d'une lettre,d'une note,d'un rapport,d'un programme, d'une convocation à une assemblée générale ou d'un devis,un texte peut donner lieu à diverses opérations.Citons à titre d'exemple : la dactylographie,la mise en page,les corrections par substitution de caractères,de mots,de phrases,la fusion,la personnalisation,l'interclassement,la sélection,la reproduction sur divers supports,la mémorisation,le classement,la recherche,la diffusion ...

Le traitement de documents intègre désormais l'ensemble de ces opérations.Ilsort donc largement du cadre de l'édition du courrier répétitif [*] et de l'aide aux corrections de dactylograhes,auquel il est resté longtemps confiné.Ce renouveau est du à l'apparition d'une nouvelle génération de matériels plus performants et en pleine évolution.

Le concept de traitement de documents a ainsi élargi son domaine à l'ensemble des opérations pouvant être exécutées sur ou à partir de textes,depuis leur saisie par la frappe manuelle jusqu'à leur restitution sur un support papier,vidéo ou magnétique.

2. LES OBJECTIFS POURSUIVIS.

Pourquoi le traitement de documents ?

Les raisons en sont économiques et technologiques.Tout d'abord,il convient d'avoir présent à l'esprit que dans la plupart des pays européens, plus de 50% de travailleurs exercent leur emploi dans un bureau.

Dans le même temps,on constate que le coût en salaires et en charges de ce personnel administratif s'accroît.Or, simultanément à cette augmentation du coût de la main d'oeuvre,celui des ordinateurs et du stockage diminuent,tandis que la qualité des imprimantes et des écrans s'améliore sans cesse. de plus en plus grande.

L'arrivée sur le marché européen des systèmes de traitement de documents à écran a coïncidé avec la prise de conscience par les employeurs,de cette augmentation des coûts administratifs alors que l'on entrait dans une période de stagnation économique.Pour beaucoup,leurs objectifs (cfr tableau 1) étaient une progression de leurs activités au moins égale à celle des années précédentes,mais "à budget constant".Ils ont commencé par chercher à réduire leurs coûts administratifs ou,ce qui revient au même,à augmenter la productivité de chaque employé et cela sans trop dégrader les conditions de travail du personnel,et des secrétaires en particulier.

objectifs des chefs d'entreprise	objectifs des secrétaires
- augmenter la productivité des traitements administratifs en particulier des travaux dactylographiques.	- travailler plus vite et dans des conditions plus agréables.
- produire plus vite l'ensemble des textes dactylographiques de l'entreprise.	- dégager un temps pour les tâches les plus importantes.
- rationnaliser les documents administratifs.	- laisser à la machine le travail répétitif et fastidieux.
- personnaliser le service rendu à la clientele (au niveau des communications écrites).	- cesser de recommencer le même document.
- alléger les structures.	- faire partir à coup sûr le courrier de dernière minute.
- garantir la correction acquise des textes.	- améliorer la mise en page (souci du "travail bien fait").
- pouvoir compter sur une évolution souple et progressive du système en fonction de l'évolution des besoins.	- disposer d'un outil de travail simple et pratique.
- améliorer la qualité des documents produits.	

tableau 1

Naissance du traitement de documents.

La vitesse de frappe d'une dactylo est un paramètre que l'on peut considérer comme fixe : la meilleure des secrétaires ne tape jamais guère plus de 60 mots à la minute.

Améliorer cette cadence de frappe a été l'objectif premier auquel se sont attachés les fabriquants de machine à écrire.Ainsi,celles-ci se sont vues dotées de fonctions complémentaires câblées ou programmées.Ces automatismes concernent par exemple les retours de chariot en fin de ligne ou de paragraphe,les tabulations,le soulignement,le centrage des titres...

D'autre part,un allégement du volume d'informations à frapper a été obtenu par l'introduction de mémoires plus ou moins étoffées selon les catégories de matériel et permettant la réutilisation des textes ou des parties de textes déjà frappés.

L'amélioration de la cadence et la réduction du volume de frappe

entraînent :

1. un **accroissement de la capacicité de production** des postes de dactylographie,mais ce n'est pas là le seul objectif poursuivi par des matériels de traitement de documents qui visent également :

2. l' **amélioration des conditions de travail,par :**
   - une diminution de la tension nerveuse des opératrices,suppression de l'appréhension de la faute de frappe en fin de texte et de corrections en fin de journée;
   - une réduction des tâches fastidieuses procurant un plus grand intérêt au travail : frappe unique,facilité de correction,...
   - un aménagement plus rationnel du poste de travail;

3. une **élargissement des services rendus,par :**
   - la réduction des délais d'exécution,
   - une meilleure présentation des documents,
   - une plus grande fiabilité de l'information tout particulièrement dans le cas de reproduction d'un original déjà contrôlé,
   - l'exécution de fonctions nouvelles,telle la sélection automatique de fichier;

4. une **réduction des coûts** ,au niveau du poste de dactylographie,mais également au niveau des rédacteurs : la possibilité de fusion et d'édition automatique de paragraphes standards préenregistrés réduit le temps de la dictée, la relecture est limitée aux seuls éléments corrigés...

3. CHAMP D'APPLICATION.

Par leurs caractéristiques techniques et fonctionnelles,les matériels de traitements de texte peuvent s'adapter aux différentes activités de **divers secteurs économiques** (cfr tableau 2).Ils concernent tout aussi bien les grandes entreprises,les banques,les compagnies d'assurances,les administrations,que les petites et moyennes entreprises qu'elles soient fabricantes ou prestataires de services.

Les applications du traitement de documents	
Secteur	Application
commercial	- publipostage - gestion de catalogues - propositions/devis - gestion de fichiers clients
secrètariat	- pool dactylographique
personnel	- recrutement - formation
finances	- budget - comptabilitè client
Production	- gammes - notices techniques
informatique	- gestion de la documentation
achats	- commandes - suivi
Etat-civil	_ extrait d'actes de naissance

Figure 2

Il convient d'èlargir le champ des applications du traitement de documents à la communication des informations,à leur archivage et à leur accès. La communication intèresse toute forme de transmission de documents,d'un système de traitement de documents à un autre,directement ou par la liaison tèlèphonique :

- la tèlècopie (courrier èlectronique);
- le tèlex;
- les systèmes de messagerie èlectronique;
- la transmission sur rèseaux informatiques spèciaux;
- la tèlèconfèrence assistèe par ordinateur.

L'**archivage de documents** "papiers" peut se faire sous forme magnètique sur des cartes,bandes, cassettes ou disquettes et disques comme la plupart des systèmes modernes de traitement de documents.Les documents peuvent ègalement être archivès sous forme micrographique - microfilms ou microfiches - ou encore sous forme d'images enregistrèes.Dans tous les cas,l'archivage implique la recherche et l'accès aux documents,source de nombreuses pertes de temps et ... de coûts prohibitifs.

Les **documents** concernès par le traitement de documents sont aussi bien ceux qui nècessitent peu de seconde frappe,comme les

lettres courtes ou les mêmos internes,que ceux où le taux de seconde frappe est élevé : rapports,études,contrats,devis,documents techniques et publicitaires... De la même manière,les documents comportant une forte standardisation, comme les propositions,les contrats ou les relances,ainsi que ceux requérant des dispositions compliquées,comme les bons de commande ou les bons de livraison,sont des candidats tout désignés au traitement de texte.

Les Personnels concernés sont,non seulement les dactylographes et les secrétaires mais également les auteurs-rédacteurs spécialisés ou cadres de tous les services.

4. CONFIGURATION GENERALE.

Il existe toute une gamme de moyens allant de la simple machine à écrire automatique qui dispose d'une petite mémoire de travail et de quelques fonctions câblées,jusqu'aux matériels les plus évolués qui correspondent tous à de véritables ordinateurs.On retrouve donc dans ces derniers les unités informatiques habituelles :

- l'unité de saisie des informations;
- l'unité logique de traitement couplée aux mémoires internes de travail
- les mémoires secondaires
- les unités de restitution

l'unité de saisie des informations.

Il s'agit essentiellement d'un clavier dactylographique.Ce dernier comporte généralement trois groupes de touches :

- un clavier alphanumérique (azerty ou qwerty)
- un clavier décimal destiné à l'introduction des données numériques,...
- un clavier de fonctions comportant les touches de commande de fonctions comme le retour en arrière,la tabulation,...

L'entrée des informations dans le système est assuré par l'opérateur qui frappe sur le clavier :

- les ordres d'identification de mise en page des textes;
- le texte lui-même et les ordres de composition qui lui sont nécessairement intégrés;
- les ordres de correction

au moment de la frappe;Et après la frappe (corrections d'auteur),

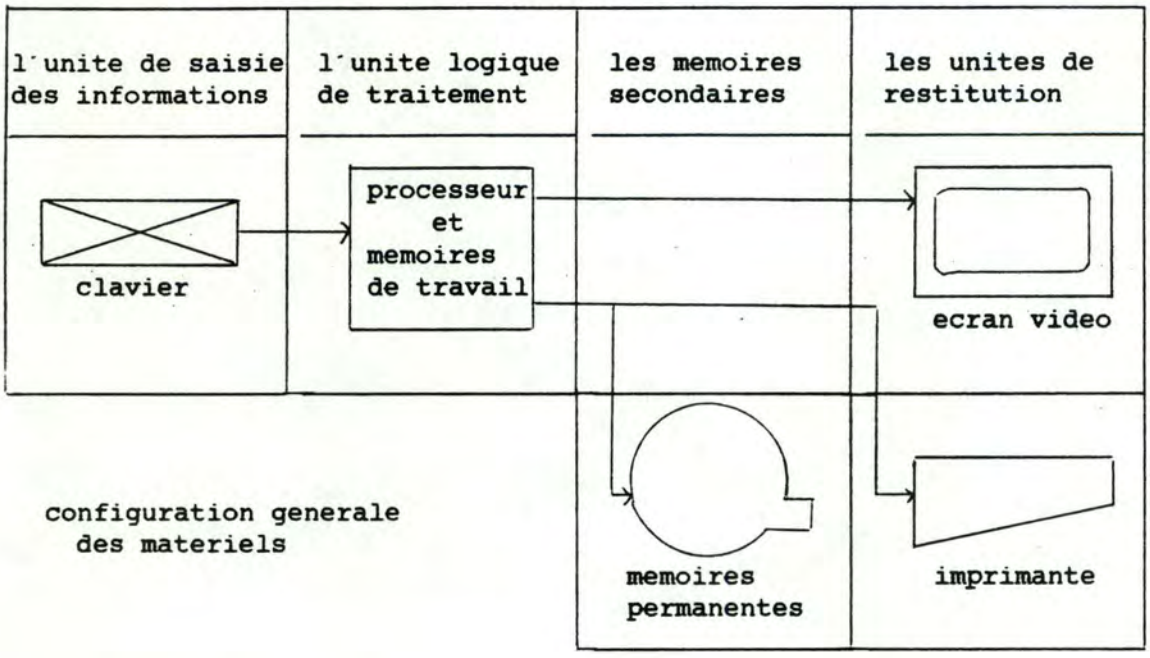
- les demandes de travaux divers sur les textes enregistrés;
- les ordres de restitution (visualisation,impression,...)

l'unité logique de traitement et les mémoires internes de travail.

L'unité de traitement et les mémoires internes constituent le passage obligé des données.

Ces mémoires reçoivent :

- les informations de base en provenance du clavier;



configuration generale des materiels

- les textes ou portions de textes à traiter en provenance de la mémoire permanente de stockage;
- Les programmes;
- les zones de travail.

Les traitements exécutés au niveau de l'unité logique et les mémoires de travail concernent :

- la correction des textes;
- les sélections;
- les fusions;
- les interclassements;
- ...

Les informations déjà mémorisées qui doivent subir des modifications sont également appelées dans les mémoires de travail où elles subissent les traitements souhaités avant d'être renvoyées dans les mémoires permanentes.

Les mémoires secondaires.

Tous les textes susceptibles d'utilisations ultérieures répétées sont stockés en bibliothèque sur des supports divers, la plupart magnétiques et immédiatement accessibles par le matériel de traitement de textes : disques, bandes, cartes, disquettes, cassettes.

Les capacités de ces mémoires varient selon le type de matériel envisagé et vont de quelques milliers à plusieurs millions de caractères.

Sont également stockés dans les mémoires permanentes:
- les ordres d'identification,de mise en page et de composition de textes;
- les inventaires de documents mémorisés;
- les programmes.

Les unités de restitution.

Les fonctions de restitution couvrent :
- la visualisation sur écran vidéo;
- l'impresion des textes,fichiers ou inventaires,contenus dans les mémoires de travail ou permanentes,sur papier continu ou feuille simple en nombre d'exemplaires souhaités.

Les écrans vidéo permettent d'afficher:
- les informations d'identification du texte;
- les informations relatives au traitement en cours;
- les inventaires de documents en bibliographie.

Les imprimantescaractère par caractère,marguerite,à jet d'encre, ,matrice ou laser sont [*] plus ou moins rapides et plus ou moins banales quant aux polices de caractères admises.

5. CARACTERISTIQUES FONCTIONNELLES.

5.1. classification des materiels.

La typologie la plus couramment retenue aujourd'hui en matière d'équipement de traitement de documents est issue de deux critères principaux de classement.Ces deux critères distinguent respectivement :

- les **systèmes autonomes** (ou monopostes) des **systèmes à logique partagée** (ou multipostes) dont le concept ,comme leur nom l'indique,a pour objet de partager les ressources d'une unité centrale (logique,capacité de stockage sur support magnétique,périphériques d'impression ou autres) entre plusieurs postes de travail clavier/écran;

- les **systèmes à écran de visualisation** dans lesquels le poste de travail comporte,en position centrale,le clavier et l'écran,l'unité d'impression étant située en périphérique, des **systèmes sans écran** dont la machine à écrire de la station de travail sert à la fois d'organe d'entrée (visualisation des données entrées) et d'unité d'impression.

Il en résulte trois principaux types de matériel :

- les **systèmes autonomes sans écran** ou machines à écrire à mémoire,ou encore machines à écrire avec commande automatique sans écran),
- les **systèmes autonomes avec écran** et
- **les systèmes à logique partagée.**

Cette classification ternaire ne concerne cependant que les matériels spécifiquement destinés au traitement de documents,c-a-d strictement conçus pour assurer les quatre fonctions de base dans ce domaine : saisie, traitement,mémorisation et impression.

En dehors des machines spécialisées,il existe des logiciels de traitement de texte pratiquement sur tous les ordinateurs exploités en temps partagé,et même sur des micros.

5.2. Les fonctions couvertes : étude comparative de certains systèmes.

Les premieres machines à écrire à mémoire n'autorisaient que la simple restitution automatique du texte frappé.Aujourd'hui des matériels de catégories supérieures exécutent des traitements plus complexes,telles que la fusion automatique de partie de textes,la substitution automatique d'un mot par un autre,...

Une étude de "bmb" [bmb 82] offre un panorama détaillé de l'offre nationale en matière de traitement de textes.Celle-c apporte les différentes réponses posées aux fournisseurs à propos de leurs modèles par le biais de trois volets distincts:

- les fiches pratiques (technico-commerciale);
- les organes du systèmes (écran et imprimante);
- les aspects fonctionnels (fonctions réalisables).

Ces trois parties donnent lieu à trois types de tableau qui permettent de comparer ces systèmes sous différents angles.

Les fiches pratiques proposées se décomposent en quatre modules :

- la fiche d'identité;
- la configuration minimale;
- la configuration maximale;
- le coût des extensions.

Les organes du système regroupent les caractéristiques techniques de l'imprimante et de l'écran.

Après avoir fait le point sur les aspects technico-commerciaux des différents modèles proposés grâce aux deux premiers volets, l'accent est mis sur les aspects fonctionnels.Les différentes fonctions reprises dans le tableau sont :

- dialogue avec l'utilisateur(français,anglais,néerlandais);
- sommaire automatique;
- glossaire (contient des termes complexes fréquents);
- fonctions de recherche;
- recherche et remplacement;
- possibilité de tri;

- tabulation décimale;
- Les quatre opérations;
- autres opérations;
- césure automatique;
- fusion de constantes et de variables
- correction orthographique;
- modifications;
- pagination automatique;
- mention bas de page;
- soulignement automatique;
- gras automatique;
- justification à droite;
- possibilités de centrage;
- traits verticaux;
- autres possibilités graphiques;
- contrôle de format;
- remise en page;
- aide pour pré-imprimés;
- marge variable;
- indices et exposants;
- retour bas de page;
- protection par mot de passe;
- communications;
- remplacer X par X.

Chapitre 2: ETUDE DES METHODES DE FORMATAGE

1. LE CADRE LIMITATIF.

Le concept "traitement de documents" couvre de nombreuses opérations et assure diverses fonctions.Elles sont utilisées dans différents domaines suivant l'objectif fixé et le matériel utilisé.

Parmi ce choix de traitements,nous allons nous attacher à un point plus spécifique,l'édition et le formatage [1] de documents techniques, de manuscrits scientifiques.Par la suite,nous nous en tiendrons à ce cadre limitatif.

2. LA PREPARATION DE DOCUMENTS.

2.1. Définition.

Un **document** peut regrouper du texte,des équations,des tableaux,des diagrammes,des photographies ,des graphiques,... Pour véhiculer l'information,les exigences d'un manuscrit scientifique sont nombreuses : c'est un document très structuré qui utilise une variété d'alphabets et de polices[*] ...

la **préparation d'un document** exige deux travaux principaux :

1. **l'édition** c-a-d la définition du contenu logique et de la structure d'un document,et
2. **le formatage** c-a-d l'élaboration du document désiré à partir des spécifications données relatives à la forme.En d'autres mots,cela concerne la présentation physique d'un document sortant sur imprimante ou sur vidéo.

A titre d'exemple,on trouvera différents types de documents prêts à être formatés dans la suite du chapitre.

Quoique le traitement de documents,spécialement l'édition,ait longtemps été la seule application des ordinateurs dans le domaine de la bureautique,ce n'est que récemment qu'une attention plus particulière a été apportée aux systèmes de formatage.La raison en est,comme nous l'avons vu au chapitre précédent,à la fois technologique et économique.

2.2. Les étapes.

Le traitement de documents se déroule en plusieurs étapes (cfr figure 1).Au départ,nous avons à l'esprit ou sur papier une

---

[1] Ces deux termes seront définis au paragraphe suivant.

image du document que nous désirons obtenir.Appelons-le **document imaginaire** Celui-ci subit une première transformation grâce à une étape d'édition,appelée **spécification.** Il en résulte le **document de description,** une forme physique,mélange de spécifications et de textes.Vient ensuite une étape de **formatage** qui fournit le **document formaté** et puis une étape de **visualisation** produit le **document concret** ,soit sur papier,soit à l'imprimante.

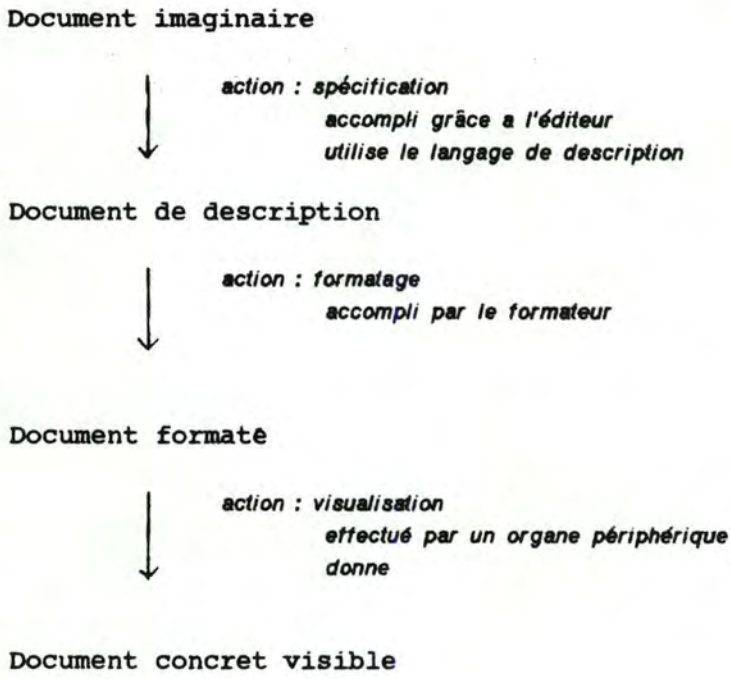


figure 1

2.3. Quelques exigences et difficultés liées à la conception de systèmes de préparation de documents.

L'utilisation des systèmes de préparation de documents pose certaines exigences.Pour un système donné,il faut

- tout d'abord,avoir le document de sortie imprimé de bonne qualité,
- ensuite,introduire le texte et les commandes de formatage de manière simple et aisée;
- enfin,avant d'imprimer un document,avoir à l'écran une vue aussi fidèle que possible de celui-ci.

Face à ces exigences,la conception des systèmes de préparation de documents présente certaines difficultés,notamment la prise en compte de la structure logique d'un document et la complexité des documents scientifiques.

La plupart des machines de traitement de documents

travaillent sur des entitès: caractère,mot,ligne,quelquefois paragraphe ou page.Mais si ces entitès permettent effectivement la représentation du texte,elles ne décrivent pas complètement un document.En plus de l'aspect physique constitué par un assemblage de ces entitès,un document a une structure logique et le système qui le manipule,pour être efficace,doit prendre en compte aussi bien le texte que cette structure.

Par exemple,du point de vue de sa structure, le présent rapport est constitutè d'une couverture,d'une table de matières,du corps du rapport ,des rèfèrences bibliographiques et des annexes.La couverture comporte le titre,le nom de la faculté,... Le corps est divisè en chapitres,eux-mêmes divisès en sections et sous-sections de différents niveaux.Chaque section est formèe d'un numèro indiquant sa position dans la structure,d'un titre et d'un corps,lui-même formè de sections de niveaux inférieurs.Au niveau le plus bas,le corps des sections est une suite de paragraphes et de figures.C'est ce type de structure que le système doit pouvoir manipuler pour assister efficacement l'auteur.

Un autre objectif de certains systèmes de traitement de documents est de prendre en compte la spècificitè des documents scientifiques.Outre le fait qu'ils sont gènèralement fortement structurès,ces documents ont la particularitè de contenir de nombreuses formules mathèmatiques ce qui implique,d'une part l'utilisation frèquente de plusieurs jeux de caractères (grec,italique,symboles mathèmatiques,...),d'autre part,la prise en compte du fait que les expressions mathèmatiques s'ètendent sur deux dimensions.Une expression de la forme

$$\lim_{x \to \pi/2} (\tan x)^{\sin 2x} = 1$$

exige un mèlange de caractères romains,italiques et grecs.L'exposant implique la prise en compte des deux dimensions de l'expression.Un exemple plus compliquè serait

$$a_0 + \cfrac{b_1}{a_1 + \cfrac{b_2}{a_2 + \cfrac{b_3}{a_3 + \ldots}}}$$

Outre les expressions mathèmatiques,un document scientifique peut contenir des tableaux.Il faut,dans ce cas dècider du placement des colonnes afin de tenir compte de la largeur des différentes entrèes de la table.
Mais on peut encore trouver dans un document beaucoup d'autres objets comme des figures,des dessins,... que l'on dèsire intègrer dans le document au cours de son èlaboration.

3. QUELQUES SYSTEMES DE FORMATAGE.

Comme nous l'avons vu précédemment,la préparation d'un document comprend deux travaux : l'édition et le formatage.C'est à ce dernier que nous nous attacherons plus particulièrement dans ce paragraphe.Tout d'abord,nous verrons quels sont les différents types de formatage.

Dans les trois paragraphes suivants,nous décrirons des formateurs purs, suivis ensuite par les éditeurs/formateurs intégrés.On trouvera ensuite quelques systèmes qui n'entrent dans aucune de ces deux catégories mais qui abordent des problèmes uniques ou qui proposent des solutions ambitieuses.Finalement,on parlera d'un certain nombre de recherches en laboratoire qui tentent de fournir des systèmes regroupant les caractéristiques les meilleures des formateurs purs et des éditeurs/formateurs intégrés.

3.1. Les différents types de formatage.

Les systèmes de formatage peuvent être divisés en deux groupes : les formateurs purs et les éditeurs/formateurs intégrés.

Les **formateurs purs** acceptent le document de description c-a-d le texte et les commandes, préparé précédemment par un système d'édition séparé.Le document formaté peut alors être visualisé,produisant ainsi le document concret visible.(Ces formateurs sont parfois appelés des compilateurs de documents ou des formateurs mode-batch).

Les **éditeurs/formateurs intégrés** permettent de visualiser le document de description,sans quitter l'éditeur/formateur.En d'autres termes,l'édition ,le formatage et la visualisation sont combinés dans un seul système. Sous sa conception la plus générale,l'utilisateur manipule directement une représentation exacte du document concret visible.Une forme ressemblant un peu plus aux formateurs purs permet de voir occasionnellement le document concret visible,ceci seulement sur demande.(Les éditeurs/formateurs intégrés sont aussi connus sous le nom d'interpréteurs de document et de formateurs interactifs).

3.2. La première génération de formateurs.

Tout au long des ans les formateurs purs ont évolué. Les premiers formateurs purs vraiment connus apparaissent en 1960.Ils disposent de périphériques relativement limités : l'organe de sortie est une simple imprimante et l'organe d'entrée est un lecteur de cartes.

Le document de description consiste en un mélange de commandes de formatage,de spécifications de dispositions typographiques et de texte.Cette forme se reflète encore dans les récents formateurs purs.Pour distinguer les commandes,du texte,il existe deux options différentes.Le premier différentie les lignes

de commandes et les lignes de donnees en mettant un caractère particulier

CALL FOR PAPERS

The aim of this conference is to survey the state of the art of computer aids for document preparation.

Papers are solicited on

- Picture editing
- Text processing
- Algorithms and software for document preparation and other related topics

Detailed abstracts should not exceed five pages; they *must* be sent before October 31, 1980 to the Program Chairman. Selected authors will be notified by November 30.

Duration of one presentation will be of either 25 or 45 minutes.

Figure 2. A sample document. Document descriptions specifying this document are presented in later figures.

en colonne 1 des lignes de commandes (p ex le point devenu bien connu avec le système RUNOFF [FURUTA 82]).La seconde option (utilisée dans le système FORMAT [FURUTA 82]) est de faire précéder chaque commande d'un caractère réservé *escape*. La fin de la commande est marquée,soit par un autre délimiteur,soit, si les commandes sont de longueur fixe,lorsque le nombre approprié de caractères a été rencontré.

Les figures 2 et 3 contiennent les documents de description utilisés par les formateurs RUNOFF et FORMAT pour produire le texte de la figure 1. Ils illustrent bien ces deux choix possibles.

Quoique ces premiers formateurs soient assez inflexibles,certaines de leurs caractéristiques ont été adoptées dans de nombreux systèmes. C'est le cas,par exemple,de

- l'entrée à deux modes,mélange de texte et de lignes de commandes;
- l'imbrication de commandes à l'intérieur du texte avec l'utilisation d'un caractère escape qui signale à partir du texte le passage au mode commande;
- l'apparition du traitement d'entité logique,comme p ex la commande paragraphe qui ne provoque pas d'action instantannée mais établit des valeurs pour des attributs généraux du document (p ex la position de la marge de gauche,la longueur de la page,...);
- l'apparition d'une commande "dictionnnaire" qui fournit une liste alphabétique des mots utilisés dans le document.

```
.center
CALL FOR PAPERS
.space 2
The aim of this conference is to survey the state of the art of
computer aids for document preparation.
.nojust
.space 1
Papers are solicited on
.space 1
.indent 10
.undent 2
- Picture editing
.space 1
.undent 2
- Text processing
.space 1
.undent 2
- Algorithms and software for document preparation and other
related topics.
.indent 0
.space 1
.adjust
Detailed abstracts should not exceed five pages; they must be
sent before October 31, 1980 to the Program Chairman.  Selected
authors will be notified by November 30.
.space 1
Duration of one presentation will be of either 25 or 45 minutes.
```

Figure 3. Document description for RUNOFF to produce the document of Figure 2. Command lines begin with a period (.). The other lines are text lines. Since there was no significant blank character in RUNOFF (unpaddable space character), ".nojust" is invoked before the itemized listing to prevent extra spaces from being inserted into the lines. Filling of lines continues. ".adjust" restores justification. ".indent" resets the left margin, ".undent" decreases the ".indent" for the one line following. The underlined word in the next to last paragraph would have to be produced by the editor (TYPSET) before running RUNOFF since RUNOFF did not have facilities for underlining.

```
)V
INDENTATION OF COLUMNS ON LEFT AND RIGHT IS (0,0),(5,0)
PARAGRAPH INDENT IS 0 PRINT POSITIONS
SEPARATION LINES BETWEEN PARAGRAPHS ARE 1
TABS ARE SET AT RELATIVE COLUMN POSITIONS 5
NONTRIVIAL BLANK IS REPRESENTED BY SPECIAL CHARACTER 44 (#)
GO
)M¢ CALL FOR PAPERS )M¢LLP ¢THE AIM OF THIS CONFERENCE IS TO SURVEY THE STATE OF
 THE ART OF COMPUTER AIDS FOR DOCUMENT PREPARATION. )P ¢PAPERS ARE SOLICITED ON
)LLH2W1 ##- )T ¢PICTURE EDITING )HLLH2W1 ##- )T ¢TEXT PROCESSING )HLLH2W2 ##- )T
 ¢ALGORITHMS AND SOFTWARE FOR DOCUMENT PREPARATION AND OTHER RELATED TOPICS. )HP
 ¢DETAILED ABSTRACTS SHOULD NOT EXCEED FIVE PAGES; THEY )U MUST )U BE SENT BEFOR
E ¢OCTOBER 31, 1980 TO THE ¢PROGRAM ¢CHAIRMAN. ¢SELECTED AUTHORS WILL BE NOTIFIE
D BY ¢NOVEMBER 30. )P ¢DURATION OF ONE PRESENTATION WILL BE OF EITHER 25 OR 45 M
INUTES.
```

Figure 4. Document description for FORMAT to produce the document of Figure 2. The lines following the ")V" until the line containing GO are paragraph-level commands, defining global attributes which hold until they are reset. Each symbol within the text following the escape symbol ")"and preceding the next blank is a phrase-level command which has a more limited scope of action. Some, such as ")P", the begin paragraph command, and ")L", the terminate current line command, have an immediate effect. Others, such as ")M", center phrase, and ")¢", capitalize phrase, serve as toggles. The first appearance turns the action on, the next turns the action back off. Character-level commands, represented by special symbols (¢, capitalize next character, and #, significant blank, in this document) affect the next character only. Notice that ¢ is both a phrase-level and a character-level command. Input is expected to come from cards. Characters are converted to lower case unless a "capitalize" command is in effect. The end of a line has no special significance within the input.

3.3. Les premiers formateurs structurés.

Entre 1960 et 1970 se développe une nouvelle génération de formateurs (les premiers formateurs structurés) basée sur les leçons tirées de l'utilisation de la première génération.Superficiellement le document de description semble identique.Cependant,les fonctions utilisées sont plus nombreuses et plus sophistiquées.Des macros permettent de rassembler des séquences de commandes souvent utilisées,de définir de nouvelles commandes et de refléter la structure logique du document lors de son introduction.Des structures de contrôle conditionnelles,des expressions arithmétiques,des variables sont empruntées aux langages de programmation fournissant une structure dans l'introduction d'un document. Les formateurs deviennent plus faciles d'emploi grâce à l'apport de certaines aides : numérotation automatique de section,table de matières,...

C'est dans ces premiers formateurs structurés qu'apparaît l'idée que formater un document n'est pas seulement prendre une suite de mots ,les mettre dans des lignes et imprimer celles-ci sur une page.Au lieu de cela,le document est vu comme un ensemble d'entités logiques (phrases, paragraphes,sections) et le but du formateur est de permettre la manipulation de ces entités.

Une autre différence entre les formateurs de première génération et les premiers formateurs structurés est l'amélioration de l'environnement du traitement de documents et des unités périphériques.Néanmoins,le formatage traite encore les différents organes de sortie.La visualisation et le formatage sont encore contenus dans le même package.

C'est dans cette catégorie de formateurs,que se situe NROFF,le système de formatage de UNIX.La figure 4 reproduit le document de description utilisé par ce formateur pour produire le texte de la figure 2.

3.4. Les formateurs structurés avec de nombreux outils.

Dans ce paragraphe,nous aborderons trois des plus intéressants et des plus influents systèmes de formatage purs d'emploi courant : Scribe,TEX et l'ensemble des outils de formatage donnés par la version 7 de UNIX [FURUTA 82][KNUT 79][KERN 75][KERN 78][KERN 81][LESK 76].Ces systèmes pourraient être appelés "formateurs structurés avec de nombreux outils" vu la sophistication et la flexibilité de ces systèmes sans cesse croissantes et particulièrement en ce qui concerne la définition d'entités logiques à l'intérieur du document.

Ces systèmes sont beaucoup plus **fonctionnels** que les premiers formateurs purs.TEX et le système UNIX peuvent inclure des équations mathématiques compliquées dans leurs documents (cfr figures 5,6,7).La spécification des tables dans UNIX est particulièrement aisée. Plusieurs objets peuvent être intégrés,particulièrement dans le système UNIX,qui permet d'inclure des équations mathématiques dans des tables et,depuis

peu,de dessiner dans du texte,des figures qui elles-même peuvent contenir

```
.ll 70
.ce 1
CALL FOR PAPERS
.sp 1
The aim of this conference is to survey the state of the art of
computer aids for document preparation.
.sp 1
Papers are solicited on
.sp 1
.in +5
.ti -2
-\ Picture editing
.sp 1
.ti -2
-\ Text processing
.sp 1
.ti -2
-\ Algorithms and software for document preparation and other
related topics.
.in
.sp 1
Detailed abstracts should not exceed five pages; they
.ul 1
must
be sent before October 31, 1980 to the Program Chairman.
Selected authors will be notified by November 30.
.sp 1
Duration of one presentation will be of either 25 or 45
minutes.
```

Figure 4. Document description for NROFF to produce the document of Figure 2. Command lines begin with ".", the remaining lines are text lines. The escape character "\" is used to give the following character special meaning. "\ ", used here, is an unpaddable space character (significant blank). ".in +5" increases the current left margin by five characters; ".in" restores it to its previous value.

With the exception of the "\ " sequence, this simple example could also be processed successfully by ROFF, NROFF's predecessor. Other mechanisms existed in ROFF to provide unpaddable spaces.

$$\frac{1}{2\pi}\int_{-\infty}^{\sqrt{y}}\left(\sum_{k=1}^{n}\sin^2 x_k(t)\right)(f(t)+g(t))\,dt$$

Figure 5. A sample mathematical equation [KNUT79c, p. 91]. Figure 6 shows specification of this equation in EQN. Figure 7 shows this equation specified in TEX.

```
.EQ
1 over {2 pi} int from {- inf} to {sqrt y}
left ( sum from k=1 to n sin sup 2 x sub k (t) right )
left ( f(t) + g(t) right ) dt
.EN
```

Figure 6. The equation of Figure 5 specified in EQN. Text enclosed in brackets, "{" and "}", is *grouped* and syntactically treated as if it were a single unit. "sub" means subscript, "sup" means superscript, "left (" and "right)" bracket a group which is surrounded by parentheses large enough to enclose the group's contents. Notice that EQN will automatically set function names, for example "sin," in a roman font instead of in the italic font used for the other textual material in the finished equation.

du texte.Chacun de ces systèmes admet une variété d'unités de sortie.Scribe fournit une description indépendante de l'unité de sortie,TEX produit un document formaté indépendant de l'unité de sortie.

La philosophie cachée derrière les interfaces utilisateurs de ces systèmes diffère assez fort.L'écart entre TEX et Scribe est le plus grand ,le système de formatage UNIX se situe à mi-chemin entre les deux. L'utilisateur de TEX est considéré comme un auteur qui veut placer les entités d'une manière très précise sur une page afin de produire un document d'aspect le plus raffiné possible.Par conséquent l'accent est mis sur le pouvoir et la flexibilité du langage de formatage.
L'utilisateur de Scribe est considéré comme un auteur plus intéressé par une spécification facile des entités abstraites à l'intérieur d'un document.Il laisse les détails de présentation des entités à un expert qui établit les découpes des entités de l'auteur sur la page imprimée.L'accent est mis sur la simplicité du langage d'introduction et les aides d'utilisation.

Chaque système comporte des détails d'organisation et d'implementation [*] intéressants.
TEX présente le formatage comme un problème d'optimisation.Pour remplir les lignes,l'intention n'est pas d'ajuster du texte aussi dense que possible dans une région mais de réduire les effets indésirables tels que les coupures de mots excessives et les fenêtres.
Scribe tente de séparer le contenu du document des actions de formatage en utilisant un langage de formatage principalement déclaratif.Scribe permet aussi de définir facilement de nouveaux environnements grâce à des modifications partielles des environnements existants.Les définitions et les déclarations globales doivent précéder tout texte.Les changements par rapport aux environnements standards peuvent dès lors être facilement détectés lors de modifications ultérieures du document.
Le système UNIX se compose de plusieurs petits programmes qui peuvent se connecter ensemble selon une variété de configurations.Son approche "construction par blocs" contraste avec celles de TEX et de Scribe qui sont toutes deux implémentées en des programmes vastes et monolithiques.

Les figures 8 et 9 illustrent ces différences : elles donnent les documents de description nécessaires pour TEX et Scribe pour obtenir le texte de la figure 1.

3.5. Les éditeurs/formateurs intégrés.

Les systèmes abordés dans ce paragraphe combinent les caractéristiques d'un éditeur de texte interactif avec celles d'un formateur de documents.Ils sont appelés les éditeurs/formateurs intégrés.

Ces systèmes se divisent en deux grandes catégories.
Dans la première,l'édition et le formatage travaillent sur les mêmes entités (mot,ligne,paragraphe,...) mais les commandes

d'édition et de formatage n'ont pas été intégrées.Nous n'avons qu'une

```
$$ {1 \over 2\pi} \int\limitswitch↓{-\infty}↑{\sqrt y}
\bigglp \sum↓{k=1}↑n \sin↑2 x↓k(t) \biggrp
\biglp f(t)+g(t)\bigrp\,dt $$
```

Figure 7. The equation of Figure 5 specified in TEX [KNUT79c, p. 149]. This specification is extremely similar in form to that in EQN. See Figure 6 and the text for discussion. "\limitswitch" causes the limits to be placed above and below the integral sign. By default (in this example, the default would have been used if the specification had been "...\int↓{−\infty}..."), limits are placed to the right of the integral sign. "\bigglp" and "\biggrp" are particular parenthesis characters somewhat larger than "\biglp" and "\bigrp", which themselves are slightly larger than the standard left and right parenthesis. Spaces have been added to improve readability, but only those separating control sequences from subsequent letters are actually required.

```
@Style(indent=0,spacing=1,spread=1)
@Heading(CALL FOR PAPERS)

The aim of this conference is to survey the state of the art of
computer aids for document preparation.

Papers are solicited on
@Begin(Itemize)
Picture editing

Text processing

Algorithms and software for document preparation and other related
topics
@End(Itemize)

Detailed abstracts should not exceed five pages; they @i(must) be
sent before October 31, 1980 to the Program Chairman.  Selected
authors will be notified by November 30.

Duration of one presentation will be of either 25 or 45 minutes.
```

Figure 8. Document description for Scribe to produce the document of Figure 2. Environment and command keywords are preceded by the escape character @, which cannot be changed, and are optionally followed by delimited arguments or delimited text. The delimiters can be just about any matched pair of brackets. The position of keywords on the input line is not significant. Paragraphs are flagged with a blank line. The @Style command modifies global attributes of the document. If a @Style command is included, it must appear at the beginning of the input file before any text is encountered. Here, the "indent" argument specifies how much a paragraph should be indented, "spacing=1" means the document should be single spaced, "spread" indicates how many blank lines should be left between paragraphs. The "@i" environment contains a text string to be italicized. An equivalent way to specify any environment with delimited string argument is to use @Begin and @End command brackets. Thus the italicized string "@i(must)" could equivalently have been specified as "@Begin(i)must@End(i)".

visualisation occasionnelle du document concret visibe (par exemple QUIDS [FURUTA 82] possède plusieurs modes de commandes dont un pour les commandes d'édition et un pour les commandes de formatage).
Dans la seconde catégorie,les entités et les commandes ont été intégrées.Les modifications d'édition se répercutent directement sur une représentation du document concret visible (ex : Bravo,Star et Smalltalk développés par Xérox et le Wang Word Processor [FURUTA 82]).On voit apparaître,de plus en plus,des écrans à fenêtres multiples qui permettent de visualiser simultanément plusieurs documents,plusieurs morceaux d'un même document ... On peut remarquer aussi que,de plus en plus,les systèmes de traitement de documents sont munis d'un dispositif de pointage tel que la souris,le crayon lumineux [*] ...

Deux observations générales ressortent de la comparaison entre les éditeurs/formateurs intégrés et les formateurs purs.

Tout d'abord,les éditeurs/formateurs intégrés tendent à plus de dépendance vis-à-vis de la configuration que les formateurs purs.Depuis le terminal classique aux écrans graphiques,l'étendue des périphériques utilisés est assez vaste.Contrairement aux formateurs purs,la création de chaque système semble subir l'influence de l'environnement dans lequel il opère.

Ensuite,les entités utilisées dans les éditeurs/formateurs intégrés sont en général moins raffinées que celles utilisées dans les formateurs purs, spécialement si on prend pour référence celles des "formateurs purs avec de nombreux outils" (notes infrapaginales,équations mathématiques, tableaux,...).Les éditeurs/formateurs intégrés n'offrent pas d' entités au niveau d'abstraction comme le permet Scribe et n'autorisent pas le contrôle minutieux de la présentation du document concret visible fourni par TEX.De nouveaux éditeurs/formateurs se développent,néanmoins dans ce sens,et utilisent aussi bien les entités concretes que les abstraites (cfr paragraphe 3.7).

3.6. Autres systèmes [FURUTA 82].

En dehors des systèmes mentionnés jusqu'à présent,il en existe d'autres qui ont des propriétés spécifiques:

- Tout d'abord,KATIB/HATTAT un formateur pur d'écriture arabe,l'alphabet probablement le plus difficile à imprimer.
- Ensuite,TRIX/RED;Ce formateur est un système de traitement de document extensible.Il permet le mélange des graphiques colorés (figure),de texte et d'équations mathématiques.
- Citons aussi IDEAL,un langage textuel de description d'objets graphiques à deux dimensions.Il utilise un système d'équations pour définir les relations entre les points significatifs de la figure objet.

3.7. Quelques développements actuels.

Actuellement,il existe,entre autres,trois systèmes expérimentaux encore en développement et pas encore complètement spécifiés : JANUS de IBM [CHAMBE 81],Etude de MIT [GOOD 81] et PEN de Yale [ALLEN 81]. Ils combinent tous l'idée d'une spécification déclarative de haut niveau de l'entité,prise dans les plus récents formateurs purs,avec l'idée d'une visualisation continue du document concret visible, comme c'est le cas dans certains éditeurs/formateurs intégrés.Ces systèmes s'inspirent tous les trois du modèle hiérarchique de Scribe pour décrire la structure abstraite d'un document et du modèle de boîtes et de colle de TEX [KNUTH 79] pour décrire les relations entre les objets concrets.JANUS et Etude tentent de fournir en même temps l'information abstraite et concrète en permettant à l'utilisateur d'éditer à la fois la structure abstraite du document (spécification du document) et son format concret (document résultant).Un des aspects intéressants de PEN est la structure en arbre pour représenter un document.Ce système a aussi développé un langage de spécification pour les formules mathématiques.Ce langage est en APL,ce qui permet des spécifications concises de certaines structures mathématiques telles que les tableaux et des suites,comme on peut le voir à la figure 10 ci-dessous.

```
\input basic % defines the standard macros, formatting parameters
\parskip 10pt
\parindent 0pt % no indentation
\def\yskip{\vskip3pt}
\def\textindent#1{\noindent
   \hbox to 19pt{\hskip0pt plus1000pt minus 1000pt#1 }\!}
\def\hang{\hangindent19pt}
\hsize 4in
\ctrline{\bf CALL FOR PAPERS}
\vskip 24pt
The aim of this conference is to survey the state of the art of
computer aids for document preparation.

Papers are solicited on
{\parskip 0pt
\par\yskip\textindent{$\bullet$}\hang Picture editing
\par\yskip\textindent{$\bullet$}\hang Text processing
\par\yskip\textindent{$\bullet$}\hang Algorithms and software for
document preparation and other related topics}

Detailed abstracts should not exceed five pages; they {\sl must} be
sent before October 31, 1980 to the Program Chairman.  Selected
authors will be notified by November 30.

Duration of one presentation will be of either 25 or 45 minutes.

\vfill % fill out rest of page with space
\end
```

Figure 9 . Document description in TEX specifying the document of Figure 2. Text following a percent sign (%) is commentary and is ignored by TEX. The first seven lines of the specification establish macros and formatting parameters. "\parskip" defines the space which is to be left between paragraphs and "\parindent" the indentation at the beginning of each paragraph. The definitions of "\yskip", "\textindent", and "\hang" are adapted from Appendix E of the TEX reference manual [KNUT79c, p. 165]. "\yskip" will leave a small amount of vertical space. "\textindent" and "\hang" will be used in specifying lists of items flagged with a bullet in the left margin. "\hsize" establishes the document's line width.

The text of the document begins with line nine. Notice the difference in line nine in syntax between a group used as an argument to a command or macro, in this case as argument to "\ctrline" which centers the argument on the line, and a group used to limit the scope of a formatting parameter, here "\bf" which switches to a bold face font. "\vskip" specifies vertical blank space. "\noindent" inhibits indentation of the first line of the following paragraph. A blank line terminates the preceding paragraph, contributing its lines to the current page; the "\par" command could have been used instead. Notice that the measurements expressed in these specifications are stated in points (a point is 0.013837 inch) and therefore are highly oriented to the visible concrete document.

PEN-MATH expression	array pattern shape	array pattern (formatted output)
n ⍳ m	m−n+1	n n+1 n+2 . . . m
n ⍳{1} m	m−n+1	n . . . m
n ⍳{4 2} m	m−n+1	n n+1 n+2 n+3 . . . m−1 m
n ⍳{0 0} m	m−n+1	n n+1 n+2 . . . m
9 ⍳ 3	7	9 8 7 . . . 3
9 ⍳{0 0} 3	7	9 8 7 6 5 4 3
⍳{4 2} ∞	∞	1 2 3 4 . . .
−∞ ⍳{4 2} 0	∞	. . . ¯1 0

Controlling array pattern *show* with index generator *looks*.

PEN-MATH expression	formatted output (resultant object)
a↑1	a^1
a↑1 2 3	$a^1\ a^2\ a^3$
a b c↑2	$a^2\ b^2\ c^2$
a b c↑1 2 3	$a^1\ a^2\ a^3$ $b^1\ b^2\ b^3$ $c^1\ c^2\ c^3$
a↓1 2 3↑1 2 3 4	$a_1^1\ a_1^2\ a_1^3\ a_1^4$ $a_2^1\ a_2^2\ a_2^3\ a_2^4$ $a_3^1\ a_3^2\ a_3^3\ a_3^4$
(a↓1 2 3)↑1 2 3	$a_1^1\ a_1^2\ a_1^3$ $a_2^1\ a_2^2\ a_2^3$ $a_3^1\ a_3^2\ a_3^3$

Examples of array generation via subscripting.

$$\begin{matrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & a_{m3} & \cdots & a_{mn} \end{matrix}$$

Array generated by: A ⍳m ⍳n

Figure 10.

4. CRITIQUES ET PERSPECTIVES.

Les paragraphes précédents ont dégagé un certain nombre de possibilités et de concepts.Ils suggèrent une réflexion sur une évolution désirable des systèmes de formatage,du point de vue du modèle qui leur est sous-jacent,des fonctions et du langage de formatage, de l'intégration des entités et des fonctions de traitement de documents ,de l'interface utilisateur et de l'implémentation.

4.1. Modèles de documents et de traitements.

Un système de formatage doit offrir aujourd'hui un modèle suffisamment explicite prenant en compte la structure logique des documents (cfr paragraphe 2.3).Dans ce domaine,la notion de "type de documents" dans Scribe et la structure en arbre de PEN s'en rapprochent fortement.

Pour décrire un document,il faut aussi tenir compte du caractère à deux dimensions de la page (équations,tables,...).Les modèles développés utilisent la juxtaposition de boîtes rectangulaires.Les plus complètes dans ce domaine sont EQN et

surtout TEX.

Certains systèmes expérimentaux tels que étude,JANUS et PEN tentent d'intégrer ces caractères concret et abstrait du document.

Pour formater les documents,les méthodes sont variées.La méthode traditionnelle consiste à accumuler des caractères dans des lignes et des pages ... Un autre modèle est basé sur l'analyse grammaticale d'un langage context-free.

Vu leur forte interaction,les modèles de documents et les méthodes de traitement doivent être développés et considérés ensemble.

4.2. Les fonctions de formatage.

Les formateurs doivent pouvoir traiter toutes sortes d'objets.Les systèmes courants fournissent des objets de type texte.Peu offrent des entités mathématiques et des tables et encore moins des histogrammes,des quartiers de tartes,des dessins.

Il y a encore beaucoup de types d'objets spécialisés et utiles qu'il faudrait formater comme les notations musicales,les diagrammes chimiques,les jeux d'échec,les mots croisés,les organigrammes, ...

De plus,les documents consistent en un mélange d'entités simples combinées à l'intérieur d'autres plus complexes (p.ex. des tableaux peuvent contenir des équations mathématiques).Il faut en tenir compte dans la structure abstraite du document.

4.3. Le langage de formatage.

L'utilité d'un système de formatage dépend fortement du langage employé pour spécifier le formatage désiré.Puisque les systèmes de formatage disposent d'un plus grand public que les systèmes de programmation conventionnels,il est important que ce langage soit facile à utiliser et à comprendre .Il est aussi important,cependant,que le langage soit capable de décrire tout document désiré.La grande difficulté du langage de formatage est de trouver un compromis entre ces deux buts.

4.4. Intégration d'entités.

Les systèmes qui traitent un grand nombre d'entités différentes omettent parfois de fournir une structure de traitement uniforme .Le système UNIX,par exemple,offre des langages et des programmes séparés pour les différentes classes d'entités (équations, tables,...).Il faudrait essayer de trouver

un ensemble simple de primitives utilisables pour crèer et manipuler des entitès de tous types.

4.5. Intègration de fonctions de traitement de documents.

Pour d'intègrer les differents types d'entitès dans une structure simple,il faut intègrer dans les systèmes les differentes fonctions utilisèes lors de la prèparation du document.Les systèmes dans lesquels ce travail n'a pas ètè fait,nècèssitent plusieurs environnements qui n'ont aucun lien entre eux.Par exemple,un environnement serait l'èditeur ,un autre l'interprèteur de commande,un troisième le système de formatage lui-même.Dans chacun des environnements,les commandes et les opèrations sont differentes.Le style d'interaction differe,même. Pour passer d'un environnement à un autre,il faut un effort mental et du temps.

Dans ce domaine,les plus grands apports viennent de l'intègration de l'èdition et du formatage dans le même système.Mentionnons,ici,LISA, le système de bureau rècemment commercialisè par Apple.Il intègre complètement tous les progiciels [*] qui ètaient auparavant disponibles sèparèment : il est ainsi possible dans un progiciel graphique,d'appeler un programme de traitement de texte : l'utilisateur dispose alors de toutes les commandes pour insèrer du texte dans des graphiques.

Afin d'être plus efficaces,les systèmes de prèparation de documents doivent offrir plus qu'uniquement des fonctions d'èdition et de formatage. Ils doivent aussi pouvoir dètecter des erreurs,gènèrer une table de matières,de concordances,des citations de rèferences bibliographiques ... De plus,lorsqu'un document devient important,les modifications ne portent que sur certaines parties qu'il faut pouvoir reformater sèparèment sans devoir reformater tout le document ...

4.6. Interface utilisateur.

Chaque système de formatage fournit à l'utilisateur un moyen d'accèder aux opèrations de crèation,de visualisation et de modification de documents.La qualitè de l'interface prèsentè à l'utilisateur peut être jugèe en partie sur les critères suivants :

- la quantitè de dètails que l'utilisateur doit mèmoriser pour utiliser le système;
- l'effort physique et mental exigè pour effectuer des commandes ordinaires;
- le nombre moyen d'erreurs faites par l'utilisateur,spècialement celles dont la correction est difficile;
- le temps d'attente de l'utilisateur suite à l'exècution de fonctions du système telles que l'adaptation du contenu d'un ècran d'un èditeur/formateur intègrè,la crèation d'un document concret à visualiser pour un formateur pur.

- le type de dialogue entre l'utilisateur et le système (clavier,souris, crayon lumineux ...)

Le développement de nombreuses techniques software a permis d'améliorer certains aspects de l'interface mais la plupart de ces compromis exigent des compromis dans d'autres domaines.Par exemple,l'utilisation de longs identifiants pour les noms de commandes réduit la quantité de détails à mémoriser par l'utilisateur puisque les noms peuvent être des descriptions de leur action.Cependant l'effort physique demandé est plus grand que si on avait un petit identifiant ... à moins que l'on ait un dispositif de pointage sur un identifiant présenté à l'écran.

L'utilisation de fenêtres multiples a réduit l'effort mental découlant du passage d'un contexte à un autre puisqu'elles permettent de maintenir plusieurs contextes simultanément.Par exemple,elles peuvent contenir des menus,différentes parties d'un même document,... Cependant,leur utilisation augmente le temps des fonctions d'édition et introduit un travail mental de corrélation entre les informations des différentes fenêtres.

Il est impossible de satisfaire simultanément toutes les propriétés souhaitables d' un interface pour toutes les classes d'utilisateurs.Cependant on pourrait offrir un interface différent suivant la classe d'utilisateurs concernée.

L'interface utilisateur peut aussi être amélioré grâce à l'emploi de techniques hardware telles que les terminaux graphiques,les possibilités de stockage ...

Des dispositifs tels que la souris,le crayon lumineux et le manche à balai facilitent considérablement la sélection et le positionnement des objets et simplifient l'approche du système.

4.7. Implémentation.

Le caractère pratique et utile d'un système de traitement de documents dépend aussi de son implémentation.A part pour le choix d'algorithmes et de structures de données de bas niveau,une implémentation peut aussi décomposer le problème de formatage en de petits morceaux indépendants et fournir une indépendance par rapport aux organes périphériques.Un autre aspect d'une implémentation est la facilité avec laquelle elle s'adapte dans un plus grand système que celui dont elle fait partie.

Chapitre 3: POSITION DU PROBLEME PROPOSE : ADJONCTION DE POSSIBILITES MATHEMATIQUES AU SYSTEME UNIX DES FACULTES.

1. LES BESOINS,LE PROBLEME POSE.

Ce mémoire a pour objet de traiter des documents mathématiques sur UNIX (version 6 du système).Le système de formatage à la disposition de la majorité des utilisateurs est le programme NROFF muni de ses jeux de macros-instructions " -MC" ou "-MVARIO".Ce programme met en page du texte mais il n'est néanmoins pas suffisant quand on veut introduire des équations mathématiques parce que,d'une part,on a des caractères,des tailles et des polices divers,d'autre part,les expressions mathématiques s'étendent sur deux dimensions.

Ce système doit tenir compte des propriétés souhaitées pour tout système de préparation de documents (cfr chapitre 2) : tout d'abord, le document imprimé doit être de bonne qualité.Pour l'obtenir,le texte et les commandes de formatage doivent pouvoir s'introduire d'une manière simple et aisée.Avant d'imprimer le document,le système doit permettre d'obtenir à l'écran une image aussi fidèle que possible de celui-ci.

En dehors de ces désirs,ce système est aussi soumis à des contraintes inhérentes à l'environnement : le document résultant est imprimé sur Sanders et les terminaux utilisés pour une éventuelle visualisation préliminaire sont,soit des VT100,soit un Mico Bee 2.Les écrans disponibles ne sont donc ni graphiques,ni semi-graphiques.

Dans la suite de ce chapitre,nous décrirons d'abord les outils,aussi bien hardware que sofware,disponibles pour ce travail afin de mettre en évidence les propriétés exploitées.Nous proposerons,ensuite une solution qui tente de résoudre certains besoins qui se font sentir en matière de documents mathématiques.

2. LES OUTILS.

Pour résoudre le problème posé par les documents mathématiques, nous avons deux types d'outils : hardware et sofware.La première catégorie regroupe les périphériques disponibles par lesquels il faudra obligatoirement passer.L'autre regroupe certains programmes existant sur UNIX prêts à être exploités.

2.1. Les outils hardware.

2.1.1. SANDERS [SANDERS].

L'imprimante SANDERS MEDIA 12/7 est une imprimante à matrice de points. Elle utilise le principe de matrice pour contrôler avec précision le placement des points.Ce contrôle très précis permet à l'imprimante de produire des caractères de haute qualité.

L'ensemble des jeux de caractères qu'il est possible d'utiliser sur la Sanders sont repris en annexe 4.Les caractères produits résultent d'une ou plusieurs passages de la tête d'impression sur la ligne imprimée.

L'imprimante est contrôlée par un microprocesseur interne qui commande les fonctions mécaniques de l'imprimante et qui manipule et traite les données à imprimer.Le microprocesseur contrôle aussi les communications entre l'imprimante et l'ordinateur.

La Sanders répond à des commandes qui lui sont envoyées par l'ordinateur central.Celles-ci gèrent l'imprimante afin qu'elle effectue certaines fonctions.Pour les expoiter,il faut intégrer ces commandes dans le fichier à imprimer.Elles sont de deux types (liste complète en annexe 4):

- mises à part les commandes d'impression standards,
- la Sanders a des possibilités de déplacements horizontaux et verticaux, de tabulations;
- Elle peut aussi agir sur le format du texte imprimé
- et permet de changer de polices de caractères.

2.1.2. VT100.

Le VT100 est un terminal avec un écran de 24 lignes de 80 caractères (éventuellement 132 caractères) et un clavier disposant des touches alphanumériques habituelles ,de 18 touches de fonctions et de 4 touches de déplacement du curseur.

L'écran peut afficher quelques caractères graphiques (cfr figure 1) qui sont utilisés pour tracer les symboles racine,intégrale,crochet, sigma,...

Tout comme la Sanders,le VT100 reçoit des commandes de l'ordinateur central qui permet de tenir compte des actions

autres que l'impression d'un caractère à l'écran.Parmi les fonctions

SPECIAL GRAPHICS CHARACTERS

Octal Code	Graphic with US or UK Set	Graphic with "Special Graphics" Set
137	—	Blank
140	\	♦ Diamond
141	a	▒ Checkerboard (error indicator)
142	b	HT horizontal tab
143	c	FF form feed
144	d	CR carriage return
145	e	LF line feed
146	f	° Degree symbol
147	g	± Plus/minus
150	h	NL new line
151	i	VT vertical tab
152	j	┘ Lower-right corner
153	k	┐ Upper-right corner
154	l	┌ Upper-left corner
155	m	└ Lower-left corner
156	n	┼ Crossing lines
157	o	⎺ Horizontal line - Scan 1
160	p	⎻ Horizontal line - Scan 3
161	q	─ Horizontal line - Scan 5
162	r	⎼ Horizontal line - Scan 7
163	s	⎽ Horizontal line - Scan 9
164	t	├ Left "T"
165	u	┤ Right "T"
166	v	┴ Bottom "T"
167	w	┬ Top "T"
170	x	│ Vertical Bar
171	y	≤ Less than or equal to
172	z	≥ Greater than or equal to
173	{	π Pi
174	|	≠ Not equal to
175	}	£ UK pound sign
76	~	· Centered dot

figure 1.

offertes par le VT100,on a entre autres la possibilitè :
- de dèplacer le curseur sur l'écran,
- de changer de modes
- de passer aux caractères graphiques ...

2.2.1. Micro Bee 2.

Le Micro Bee 2 est un terminal avec un écran de 24 lignes de 80 caractères et un clavier disposant de touches alphanumériques et de nombreuses touches de fonctions.

De même que le VT100,ce terminal offre des fonctions qui permettent d'agir sur l'écran,d'afficher quelques

caractères graphiques ... Les commandes utilisées pour ces actions ne sont néanmoins pas les mêmes que celles du VT100.

Le Micro Bee 2 possède deux particularités dont nous tiendrons compte :

- Tout d'abord,il a 16 touches de fonctions programmables par un utilisateur.
- De plus,le clavier de ce terminal est AZERTY et possède,comme les machines à écrire ordinaires,des touches spéciales pour les voyelles à accent "è","à","ê","ö",... Le caractère correspondant et apparaissant à l'ecran est un caractère de contrôle;il y a donc une compatibilité à assurer à ce niveau lors de l'impression de ces caractères particuliers.

2.3. Les outils software.

Le système d'exploitation de UNIX possède des programmes qui permettent de formater des documents : NROFF est le formateur de texte de base.On peut lui ajouter une série de jeux de macros-instructions plus ou moins adaptés au style de documents à imprimer (livre,article Scientifique,...).Ce système possède aussi des préprocesseurs spéciaux pour les expressions mathématiques et pour les tableaux.

2.3.1. NROFF.

Dès qu'un utilisateur a introduit un document dans un fichier,il peut être formaté grâce à NROFF.
Certaines opérations de formatage sont exécutées automatiquement. Parmi celles-ci,la numérotation des pages,la justification et le cadrage du texte.
D'autres opérations peuvent être effectuées sur demande de l'utilisateur grâce à l'insertion de commandes dans le texte.Mais le langage fourni à ce niveau est d'un niveau très bas et difficile à utiliser.

2.3.2. Les jeux de Macros-instructions.

Afin de faciliter le travail de l'utilisateur,le système possède plusieurs jeux de macro-instructions.Ces macros-instructions regroupent les commandes de base de NROFF et sont donc d'un plus haut niveau.Elles permettent la numérotation automatique des sections,la création de paragraphes, l'insertion de notes infrapaginales,...

2.3.3. NEQN.

Pour aborder les difficultés posées par les expressions mathématiques ,le préprocesseur NEQN de NROFF propose un langage de description.Une intention de ce langage est d'être facilement utilisable pour les profanes en

mathématiques.Ce langage se veut "naturel", proche de la façon dont on lit les formules.

La spécification d'une équation est délimitée par les commandes .EQ et .EN.Les équations peuvent aussi s'insérer dans du texte si elles sont entourées de délimiteurs qu'il faudra préalablement définir.La construction des différentes expressions comporte différents mots réservés.Les espaces blancs n'ont aucune signification dans la description d'une équation.Certains mots réservés peuvent être définis ou redéfinis par l'utilisateur.Toutes ces propriétés sont illustrées en annexes 2 et 3 .

2.3.4. TBL.

Pour formater des tables,NROFF dispose du préprocesseur TBL.La description d'une table est délimitée par les commandes .TS et .TE.La première ligne qui suit .TS spécifie les différentes colonnes.Dans les tables, on peut trouver des équations mathématiques.

3. LA SOLUTION PROPOSEE.

3.1. Mécanisme général.

La solution proposée pour traiter les documents mathématiques sur UNIX comporte un formateur pur.Le processus de préparation du document est donc une activité cyclique où on raffine la description du document,on génère le document résultant et on trouve les défauts de présentation dans le document concret.Ce processus est répété jusqu'au moment où la présentation est satisfaisante.

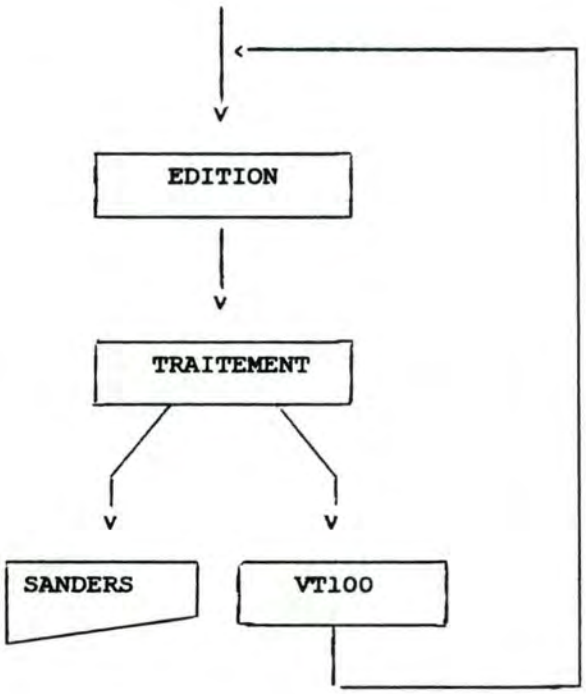


figure 2

Ce processus est schématisé à la figure 2.Pour créer et modifier le document de description,l'utilisateur peut employer n'importe quel éditeur: mince ou edit [*].L'écran utilisé pour la visualisation du document résultant est un VT100 et l'impression de ce document est lancée sur Sanders.Les paragraphes suivants exposent les possibilités du système. Afin de connaître les commandes introduites pour les réaliser,il faut s'en référer aux modes d'emploi (annexe 3).

3.2. Possibilités sur l'imprimante.

Le système permet plusieurs structures de documents : soit,le livre c-a-d une découpe en chapitres,soit l'article

scientifique.Pour ce dernier, il est possible d'avoir une présentation particulière de la première page, concernant le titre,l'auteur,l'institution,le résumé,...(cfr annexe 2). Il existe aussi des possibilités de notes infrapaginales.

Le texte proprement dit peut être écrit en simple ou en double colonne. L'utilisateur peut créer des sections,des paragraphes numérotés ou non,les énumer ... Il peut modifier les marges et les alinéas,traiter des parties de texte de manière particulière (contraction,transcription littérale,mise en évidence,...),les souligner,sauter des lignes et des pages,...

Le texte peut être écrit selon toutes les polices disponibles sur Sanders actuellement c-a-d mis à part l'écriture habituelle, LE DRAFT, *l'italique* et l'helvesan regular.
Et pour chacune de ces polices,il est possible d'écrire en **caractères gras.**

Ce système permet aussi d'insérer des formules mathématiques soit sur une ligne isolée,soit à l'intérieur même d'une ligne de texte (grâce à l'utilisation de délimiteurs définis préalablement).
Par formules mathématiques,on sous-entend des expressions contenant des indices et des exposants,des fractions,des racines carrées,des sommations, des intégrales,... Ce système peut aligner verticalement des objets,il est donc possible d'obtenir des piles,des matrices,de dessiner des grandes parenthèses et accolades,... Ce système permet aussi d'aligner des équations à un endroit donné p.ex. au signe d'égalité et connaît plusieurs signes particuliers.

La saisie se fait de façon aussi naturelle que possible à l'aide d'un langage convenu.Les différentes constructions sont obtenues grâce à l'utilisation de mots connus du système et qui décrivent l'équation (voir liste en annexe 3).Par exemple,pour obtenir $\sqrt{a}$,on utilise le mot "sqrt" suivi de a.Une série de caractères particuliers sont connus du système :

$\geq \leq \neq \pm \equiv \nabla \propto \infty \partial$

$\int \subset \supset \in \ni \not\subset \not\supset \not\in \not\ni \forall \exists \wedge \{ \} \| \cap$

ainsi que les lettres grecques :

$\Delta \Gamma \Lambda \Omega \Phi \Pi \Psi \sum \Theta \cup X \alpha \beta \chi \delta \epsilon \eta \gamma$

$\iota \kappa \lambda \mu \nu \omega o \rho \pi \varphi \rho \sigma \tau \theta u \xi \zeta$

Une dernière possibilité du système est l'établissement de tableaux. Ils peuvent contenir du texte et même des équations mathématiques. (le lecteur trouvera des exemples en annexe 2).

3.3. Possibilités sur l'écran.

Si la préparation d'un document permet d'imprimer un tel document sur la Sanders,il y a néanmoins quelques restrictions concernant la visualisation sur VT100.Ceci tient essentiellement au fait que cet écran ne possède pas ou peu de possibilités graphiques.

Contrairement à la Sanders qui permet des déplacements latéraux et verticaux quelconques,sur le VT100 le curseur se déplace horizontalement d'un certain nombre de colonnes et verticalement d'un certain nombre de lignes. Les déplacements verticaux ne se font par conséquent plus par demi-lignes mais par lignes.Les expressions mathématiques s'en trouvent donc plus allongées.

Une deuxième restriction réside dans le fait qu'on ne peut pas superposer des caractères à l'écran.Par conséquent,les voyelles à accent comme à,é,ê,ö,... ainsi que d'autres constructions comme ¥,ł,... restent incomplètes .

Enfin,ce terminal ne possède pas de signes particuliers et de caractères grecs.Ils seront remplacés par d'autres caractères pour lesquels les attributs visuels de l'écran seront modifiés.Il en est de même pour les différentes polices.

Chapitre 4: PRINCIPES DE REALISATION.

Nous avons présenté au chapitre précédent ,la solution élaborée pour préparer un document mathématique, ainsi que les outils disponibles et exploitables.

Avant d'attaquer en détails la réalisation de ce système,nous étudions et comparons deux démarches possibles (interactive ou non) pour résoudre le problème posé.Nous développons ensuite la première option afin de connaître les raisons pour lesquelles il a fallu y renoncer.Nous nous attachons,enfin,à la deuxième option et nous élaborons cette solution de manière détaillée.

1. PROPOSITIONS DE SOLUTIONS.

Dans les deux premiers paragraphes,nous donnons l'idée sous-jacente à ces deux démarches.Nous les comparons ensuite et nous déterminons les avantages et les inconvénients des deux méthodes.

1.1. Démarche interactive : Océ.

La première démarche développée est interactive.Elle s'inspire de la machine de traitement de texte Océ/CPT 8525.Ce système entièrement interactif permet,en effet,d'introduire dans un document des équations mathématiques et des tableaux .

Le principe général est le suivant : les expressions mathématiques sont entrées directement sous leur forme finale.Au lieu de les introduire comme du texte sur une même ligne,on joue avec des touches de déplacement du curseur sur l'écran,afin de positionner ce que l'on veut où on veut.
Le clavier possède plusieurs jeux de caractères.Ceci permet d'obtenir des caractères spéciaux (grec,intégrale,racine carrée,...) simplement en changeant de jeu de caractères.
L'introduction de tableaux s'opère grâce au placement de tabulations.

1.2. Démarche non interactive.

Une autre démarche possible est du type non interactif.Elle s'inspire de l'ensemble des outils de formatage donnés par la version 6 de UNIX (c-a-d nroff,ses jeux de macros-instructions,neqn,tbl).Ce système permet aussi d'introduire des tableaux dans un document (mais neqn et tbl n'ont pas encore été exploités aux facultés).

Le principe sous-jacent est le suivant : quand on tape un document contenant des expressions mathématiques,on les introduit comme une partie de texte mais on les marque par des

délimiteurs.Neqn,le prêprocesseur pour les équations mathématiques,lit le document de description et ne touche pas aux non mathématiques.En même temps,il convertit les parties mathématiques en des commandes nroff nécessaires.
Le traitement de tableaux s'opère de manière analogue;le prêprocesseur concerné est tbl.

1.3. Avantages et inconvénients.

Chacune de ces deux méthodes comporte des avantages et des inconvénients (cfr tableau 1).Remarquons qu'à part quelques exceptions, les avantages d'une méthode sont souvent les inconvénients de l'autre. Ceci est essentiellement dû au fait que les avantages qu'apporte un système ,exigent souvent certaines contreparties.

NEQN	Océ
+ Le langage utilisé pour introduire les expresions mathématiques est facilement utilisable pour les profanes en mathématiques.	+ Cette façon de procéder est très simple et facile à apprendre
+ Certaines opérations sont automatiques comme par exemple le positionnement de la barre de fraction ...	
+ Ce langage est bien adapté à l'environnement : les expressions mathématiques sont entourées par des délimiteurs et introduites comme du texte	+ Cette méthode est bien adaptée à un environnement où toute la mise en page se fait au fur et à mesure
+ la mise en page du document est très satisfaisante	- la mise en page est plus limitée quant aux polices utilisées,...
+ Pour imprimer des caractères un clavier standard suffit à leur introduction (mais sur un écran standard,il y a dans ce cas pas de possibilités de visualisation)	- Cette méthode nécessite un écran semi-graphique.
- Etant donné que les expressions ne sont pas introduites de manière interactive,on ne s'imagine pas bien la mise en page	+ Vu le caractère interactif de ce procédé,l'expression apparaît directetement telle qu'elle sera sous sa forme finale

- Lors de l'introduction de l'expression, la personne doit transposer ce qu'elle doit taper	+ Lors de l'introduction de l'expression au terminal, il n'y a aucune transposition à effectuer
- Les corrections éventuelles demandent un temps de réflexion	- le caractère interactif facilite fortement les corrections

tableau 1

Qu'en est-il des trois exigences de départ (cfr chapitre 2 et 3) ?

La méthode est-elle d'utilisation aisée? Quoique la solution interactive soit plus facile à apprendre et à manipuler, les deux méthodes sont facilement utilisables par des gens non initiés aux mathématiques.

Qu'en est-il de la qualité du document imprimé? Ces deux systèmes produisent un document final très satisfaisant. Neqn (et sûrement son extension EQN) possède néanmoins plus de possibilités quant aux polices utilisées et à la taille des caractères,...

A-t-on une visualisation aussi fidèle que possible du document? L'utilisation d'un terminal semi-graphique permet aux deux méthodes de reproduire exactement le document final. L'écran Océ permet la visualisation de la totalité d'une page imprimée. Rappelons, cependant, qu'on ne dispose pour ce travail que d'un écran standard. Il y a donc des restrictions à apporter quant à la visualisation fidèle du document (cfr chapitre 2).

2. DEVELOPPEMENT ET ECHEC DE LA METHODE INTERACTIVE.

2.1. Motivation et idée générale.

Pour traiter les formules mathématiques,deux démarches sont donc possibles.L'option interactive donne une représentation à l'écran des expressions mathématiques sous leur forme finale.Elle est donc plus lisible dès son introduction
C'est,dans cet état d'esprit,que nous allons élaborer un système de traitement des expressions mathématiques.Il faudra toutefois tenir compte que l'on ne dispose que d'un écran et d'un clavier standards.

Pour ce faire,nous disposons d'un programme de formatage "nroff" muni de ses jeux de macros-instructions.Nous savons qu'il n'est pas suffisant pour introduire des données mathématiques.
Afin d'y remédier nous allons créer un système qui les traitera.L'idée est d'introduire le texte à formater comme lorsqu'on utilisait uniquement nroff et ses macro-instructions,grâce à un éditeur orienté écran (du type MINCE) et de taper les formules telles qu'elles apparaissent sous leur forme finale.Ceci permettrait de résoudre en partie le problème bidimensionnel des expressions mathématiques.
Pour introduire tous les caractères spéciaux ne figurant pas sur le clavier standard du terminal,l'idée est d'utiliser un identificateur suivi d'une touche de ce clavier c-a-d de définir en quelque sorte un deuxième jeu de caractères (lettres grecques,$\sum$,...).

2.2. Découpe générale.

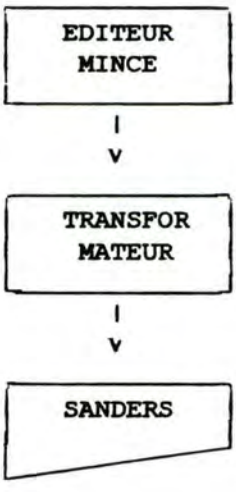


figure 1

Pour décomposer le problème,nous essayons de spécifier partiellement un programme avec non visualisation à l'écran et

réalisé selon les étapes suivantes (cfr figure 1) :
- édition avec un éditeur orienté écran
- transformation
- impression sur sanders

La transformation comprendra,comme nous le verrons ultérieurement,une phase de formatage grâce à nroff,.. et une phase de traitement des caractères spéciaux.A ce niveau apparaît déjà une contrainte: le traitement des expressions mathématiques doit être compatible avec nroff.Par conséquent,les identificateurs éventuels ne peuvent être que les caractères de contrôle spécifiques <CTRL>O ou <CTRL>N.

2.2.1. édition.

Dans un tel système,le texte est introduit par un éditeur orienté écran (comme MINCE).

Les caractères spéciaux sont identifiés grâce à une suite de caractères particuliers qui les identifient.C'est,par exemple,une expresion de la forme <CTRL>O - caractère - <CTRL>N.Pour identifier l'intégrale on utilise la séquence <CTRL>Oi<CTRL>N,la somme, <CTRL>Os<CTRL>N,...

Les formules mathématiques sont tapées directement sous leur forme finale,il faut donc les entourer par les commandes de nroff qui permettent de recopier des lignes de manière inchangée.

2.2.2. Transformation.

Une fois introduit,le texte peut être formaté grâce à nroff et ses jeux de macros-instructions.Celui-ci se contente de recopier ce qui concerne les expressions mathématiques.

Nous obtenons aloes un fichier formaté mais qui contient néanmoins des caractères de contrôle qui permettent d'identifier des signes spéciaux.Il faut transformer les séquences <CTRL>O - caractère - <CTRL>N en commandes sanders pour générer ces signes.

2.2.3. Sanders.

Le fichier est alors totalement traité,il nous reste à lancer l'impression sur la sanders.

2.3. Problèmes - Contraintes.

Malheureusement,bien que le principe de ce système ait un certain attrait, la réalisation pose de nombreux problèmes;ils

proviennent essentiellement de l'introduction du texte et de la visualisation à l'écran c-a-d essentiellement au niveau de l'édition.

Tout d'abord,il n'est pas envisageable de mettre les exposants (resp. les indices) sur la ligne supérieure (resp. inférieure) car à l'impression le décalage doit seulement être d'une demi-ligne.

Ensuite,il y a un problème de mise en page à l'édition à cause des caractères spéciaux.Pour chacun d'eux,plusieurs caractères sont introduits.Etant donné que la formule est mise en page directement, cela provoque un problème de visualisation à l'écran.En effet,quand on trace une barre de fraction,quand on écrit les bornes d'une intégrale,... on ne tient pas compte que certains caractères auront disparu à l'impression.Les barres de fraction seront donc trop longues et les bornes pas où il faut,...
Par conséquent,il faut que pour chaque caractère spécial n'apparaisse à l'écran qu'un seul caractère;sans quoi,la mise en page correcte d'une formule est impossible.

Il reste encore une question à résoudre lors de la construction de caractères. Dans le cas d'un traitement de texte avec un terminal semi-graphique, l'utilisateur construit lui-même certains caractères tels que la racine carrée,l'intégrale,... Ici,vu la non visualisation réelle à l'écran,ces symboles ne sont pas construits par l'utilisateur.Mais de quelle hauteur,de quelle largeur est la racine carrée,...?

2.4. Suggestions de solution et conclusion.

Ces problèmes essentiels restent non résolus.C'est au niveau de l'édition qu'il faut agir.

Pour les indices et les exposants,il faut exploiter le fait que dans nroff,il existe des commandes \u (monter d'une demi-ligne) et \d (descendre d'une demi-ligne).Par exemple,pour imprimer "E_2",il faut mettre dans le fichier de départ "E\d2\u". On pourrait envisager deux touches de fonction (cfr micro bee chap 2) qui écriraient dans le fichier les commandes \u et \d et qui donneraient une autre présentation à l'écran (p.ex. mettre l'indice sur une autre ligne ou utiliser les attributs visuels de l'écran).

Suite au problème posé pour la mise en page par la présence des identifiants des caractères spéciaux,on voudrait n'avoir à l'écran qu'un seul caractère,en le distinguant des caractères normaux par l'utilisation d'attributs visuels.Par exemple,le caractère grec α introduit par <CTRL>Oa<CTRL>N apparaît à l'écran comme le caractère "a" en noir sur blanc.

Afin d'élaborer une solution valable,il faudrait par conséquent un éditeur :

- orientè écran
- qui analyse les carctères introduits par l'utilisateur et qui les renvoie ou non
- qui dètecte l'emploi de touches de fonctions (ex: f_1 sur micro bee 2) et agit en consèquence à l'écran et sur le fichier
- qui interprète et exècute des commandes d'attributs visuels.

La solution èlaborée pour traiter des expressions mathèmatiques se heurte à la difficultè considérable que constitue la conception d'un éditeur.Il n'est donc pas opportun de continuer dans cette voie.

3. DEVELOPPEMENT NON INTERACTIF.

La démarche précédente s'est soldée par un échec.Il existe cependant un autre champ d'action.Puisque UNIX possède des programmes (NEQN et TBL) qui traitent des équations mathématiques et des tableaux, pourquoi ne pas les exploiter et ajouter ce qui est nécessaire à leur fonctionnement dans la configuration présente ?

3.1. Découpe générale.

Le développement de cette analyse a conduit à la solution présentée au chapitre 3.Une découpe plus raffinée des différents modules est donnée à la figure 1.

Parmi ces modules,certains sont déjà connus.Il s'agit de l'éditeur et des différents programmes de formatage tbl,neqn,nroff.
Le mécanisme détaillé de ce système se déroule comme suit:
Après l'étape d'édition,le document de description passe par un module "prefiltre".Celui-ci manipule le texte tapé par l'utilisateur afin qu'il soit formaté correctement.Il permet aussi de changer de polices de caractères et d'écrire en caractère gras.Sa raison d'être est due à la présence des deux programmes "back_space" et "ecran".Nous y reviendrons ultérieurement.A la sortie de "prefiltre", ce texte est formaté.On obtient un document formaté où les expressions mathématiques et les tableaux ont été générés.
Pour être visualisé à l'écran,ce texte passe par l'interface "écran".En plus de la visualisation immédiate,la version visualisable est enregistrée sur un fichier pour une éventuelle consultation.Les corrections peuvent donc s'effectuer jusqu'au moment où le texte est satisfaisant.
Pour imprimer le texte formaté sur la Sanders,il lui faut passer par l'interface "back_space".Il reste à lancer l'impression sur la Sanders. Celle-ci est gérée par le programme "vario".

Dans la suite de ce chapitre,nous analysons en détail les trois modules "prefiltre","back_space" et "écran".Nous spécifions les traitements effectués par chacun d'eux ainsi que leur réalisation.

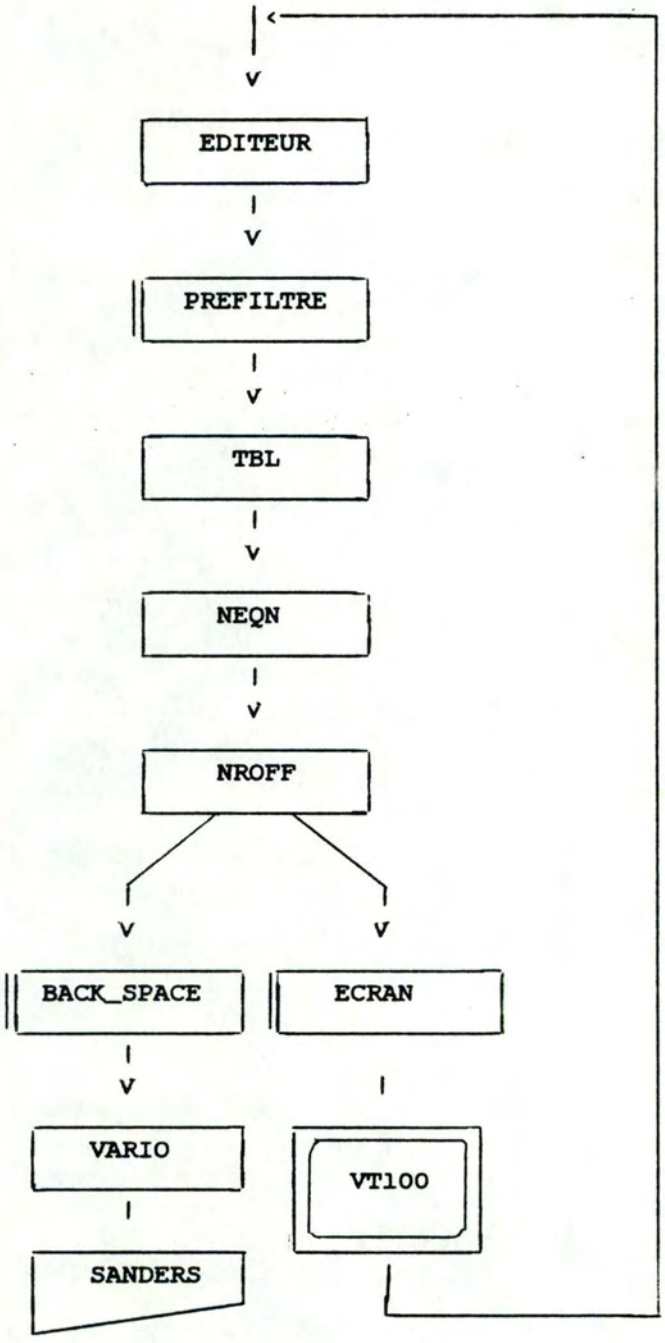

EDITEUR
PREFILTRE
TBL
NEQN
NROFF
BACK_SPACE
ECRAN
VARIO
VT100
SANDERS

figure 2

3.2. Interface Sanders : "back space".

3.2.1 Les traitements

ENTREE :

un fichier (document formatè) qui contient

- des commandes NEQN du type ‹ESC›7,‹ESC›8,‹ESC›9,‹CTRLH›, ‹CTRLN›caractère‹CTRLO› (cfr annexe 6).

- èventuellement des commandes Sanders. (èventuellement car ces commandes sont èliminées par le formatage avec NROFF).

- des commandes du type ‹CTRLO›caractère‹CTRLH›‹CTRLN› et ‹CTRLN›caractère‹CTRLO› de prefiltre.

SORTIE :

Un fichier (document formatè) dont les commandes de NEQN et celles produites par prefiltre ont èté traitées afin d'obtenir à l'impression un document dans lequel, grâce aux commandes Sanders,

- les expressions mathèmatiques sont construites de manière satisfaisante,

- les changements de polices de caractères et le passage aux caractères gras sont gènerès.

SPECIFICATIONS :

- Tout d'abord,ainsi que son nom l'indique,cet interface s'occupe des sèquences de backspaces.En effet,la Sanders ne les accepte pas.
  Lorsqu'on avait uniquement du texte,ces suites apparaissaient uniquement quand on soulignait des phrases et des mots.Les suites de backspaces ètaient suivies du même nombre de caractères "_".Le programme "vario" gèrait ce cas en supprimant les backspaces et les "_" et en ajoutant autour de la partie à souligner les commandes ‹ESC›u(n) (cfr annexe 4).
  Il n'en va plus de même lorsque le document contient des expressions mathèmatiques (p.ex. la racine carrèe ne souligne rien).Il faut donc considèrer les backspaces indèpendamment et reculer d'une certaine distance.Celle-ci est dèterminèe par les n caractères imprimables prècèdants les n backspaces (la largeur de ces caractères est variable car ils peuvent être de chasse fixe (10 ou 12 pitch) ou de chasse variable.De plus les caractères utilisès dans les commandes insèrèes à l'intèrieur du texte sont à èviter).

- Il traite aussi les "_" afin que la barre de fraction èventuelle soit positionnèe à la bonne hauteur par rapport au signe d'ègalitè.

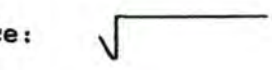
- Certains signes particuliers tels que $\leq, \geq, \equiv, \neq, \pm, =$ sont gènèrès par construction dans le système de formatage.Etant

donnè que ces caractères existent sur la Sanders,la construction est remplacée par le caractère correspondant.

- Le système de formatage gènère un deuxième jeu de caractères grâce à la séquence ‹CTRLN›caractère‹CTRLO›.Le programme "back_space" remplace cette suite par la séquence Sanders correspondant à la gènération de ce caractère.

- Il procède de la même manière pour les séquences gènèrées, non pas par neqn mais par "prefiltre".On obtient ainsi des signes spéciaux supplementaires (∩,∢,∈,...).

- La racine carrée est remaniée afin qu'elle soit obtenue en un trait continu.

NEQN :  back_space: 

- "back_space reconstruit aussi les grandes parenthèses,acccolades,crochets et barres verticales qui sont rèduits dans neqn à une suite discontinue de traits verticaux.

NEQN :   back_space: 

- Ce programme s'occupe aussi des changements de polices de caractères (normal,draft,italique,helvesan regular) et de la gènération des caractères gras.Il transforme la suite de caractères de contrôle gènèrée par "prefiltre" en la commande Sanders correspondante.

- "back_space" est compatible avec "Micro Bee".En effet, sur ce terminal,les voyelles à accent sont représentées dans le fichier par un caractère de contrôle alors que sur un terminal comme le VT100,ces mêmes voyelles sont représentées par la voyelle,un backspace et l'accent.

3.2.2 La réalisation des traitements.

La découpe choisie pour réaliser ces traitements est illustrée à la figure 3.Cette structure est conçue pour adjoindre facilement des modules supplèmentaires quand le besoin s'en fait sentir.Ce principe a été illustré tout au long du travail : tout d'abord,il a fallu imprimer les résultats obtenus par neqn (traitement des backspaces,...); ensuite,certaines amèliorations ont été apportées (ajustement de la barre de fraction,remaniement de la racine carrée,des grandes parenthèses,...);enfin,quelques possibilités supplèmentaires ont été ajoutées (caractères

gras,italiques,signes spèciaux supplèmentaires).

Les spècifications des modules intervenant dans la dècoupe sont assez volumineuses et ont ètè laissèes en annexe.Pour chacun d'eux sont dèfinis les entrèes,les sorties,l'effet,les prè-conditions, les post-conditions,les procèdures appelantes,les procèdures appelèes,les variables globales,les variables locales et l'algorithme logique.

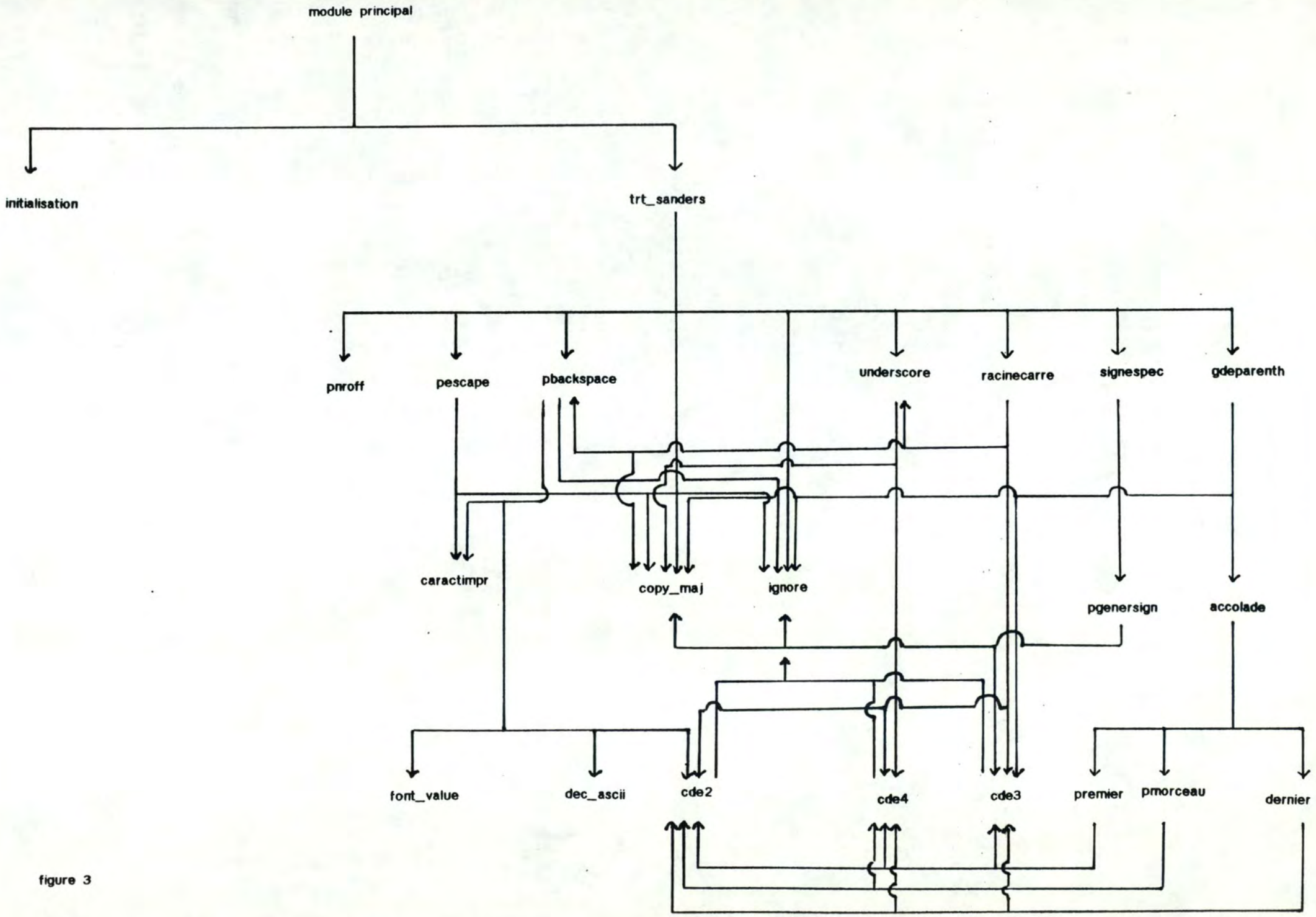

module principal
initialisation
trt_sanders
pnroff
pescape
pbackspace
underscore
racinecarre
signespec
gdeparenth
caractimpr
copy_maj
ignore
pgenersign
accolade
font_value
dec_ascii
cde2
cde4
cde3
premier
prnorceau
dernier

figure 3

3.3. Filtre préliminaire : "prefiltre".

3.3.1 Les traitements.

ENTREE :

un fichier contenant du texte,des macros-instructions de nroff,des commandes spéciales pour le préfiltre,des mots du langage de neqn.

SORTIE :

le fichier où

- on a traité les macros-instructions destinées à changer de polices et à passer aux caractères gras.

- on a posé un identificateur pour les grandes parenthèses,accolades, crochets,barres afin de les distinguer entre eux et de savoir si la parenthèse,... est droite ou gauche.

- Certains mots introduits par l'utilisateur pour obtenir un signe particulier sont remplacés par une séquence identifiante de longueur 1.

SPECIFICATIONS :

La présence du prefiltre permet d'effectuer certaines opérations qui seront utiles ultérieurement lors de l'exécution de back_space et écran :

- Tout d'abord,prefiltre transforme les macros-instructions que l'utilisateur a introduites pour changer de polices de caractères et pour obtenir des caractères gras.Il les modifie en une séquence de caractères de contrôle qui restera inchangée lors du passage dans le système de formatage et qui est de longueur nulle pour ne pas influencer ce formatage. Cette séquence de caractères de contrôle permettra à back_space et à écran d'opérer ultérieurement la transformation adéquate.

- Le système de formatage réduit les grandes parenthèeses,accolades, crochets,barres à une suite discontinue de traits verticaux.Ces parenthèses,... sont générés grâce à des mots réservés "left" et "right".On peut avoir,par exemple,"left {","right]",.... Afin que back_space et écran puissent les reconstituer,il faut permettre à ces programmes de reconnaître ces expressions.On introduira donc une séquence identifiante de caractères de contrôle devant le "left" ou le "right",de longueur nulle,non modifiée par le système de formatage.

- La Sanders peut générer plus de caractères spéciaux que ne le propose neqn.C'est donc prefiltre qui prendra en charge ces caractères. A nouveau,cette génération se fait en deux

étapes.La première est prise en charge par prefiltre et permet à back_space et à écran de savoir de quel caractère spécial il s'agit grâce à une séquence de caractères de contrôle,de longueur nulle,non modifiée par le système de formatage.

3.3.2 La réalisation des traitements.

La découpe choisie pour réaliser ces traitements est illustrée à la figure 4.
Pour percevoir cette structure,il faut savoir que la présence des expressions mathématiques est précisée,soit par les deux macros-instructions .EQ et .EN sur les lignes précédente et suivante, soit par des délimiteurs définis par l'utilisateur et les enserrant dans la ligne contenant la ou les expressions.
La principale fonction de prefiltre est de chercher des macros-instructions et des mots réservés dans une expression mathématique.Selon les lignes du fichier d'entrée,il y a trois cas à prendre en considération :

- les lignes de texte normales;

- les lignes comprises entre les deux macros-instructions .EQ et .EN;

- les lignes pour lesquelles les délimiteurs définis par l'utilisateur sont à prendre en considération.

Ces trois cas se reflètent dans la structure choisie.

Les spécifications des modules intervenant dans la découpe ont été laissées en annexe.Y sont définis les mêmes points que pour les modules de back_space.

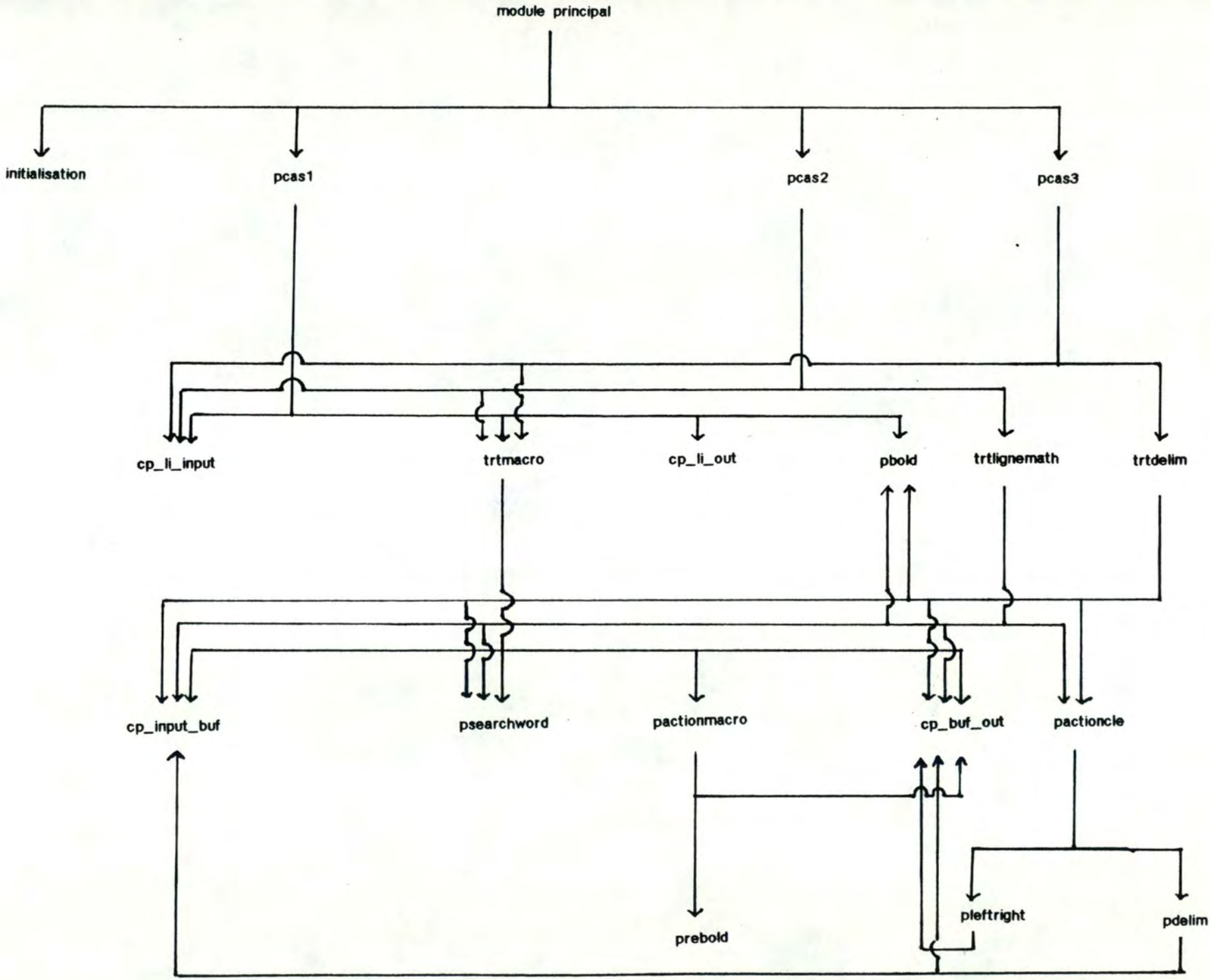

module principal
initialisation
pcas1
pcas2
pcas3
cp_li_input
trtmacro
cp_li_out
pbold
trtlignemath
trtdelim
cp_input_buf
psearchword
pactionmacro
cp_buf_out
pactioncle
prebold
pleftright
pdelim

figure 4

3.4. Interface VT100 : "écran".

3.4.1 Les traitements.

ENTREE :

un fichier (document formatè) qui contient :

- des commandes NEQN du type <ESC>7,<ESC>8,<ESC>9,<CTRLH>, <CTRLN>caractère<CTRLO> (cfr annexe 6).

- éventuellement des commandes Sanders. (éventuellement car ces commandes sont éliminées par le formatage avec NROFF).

- des commandes du type <CTRLO>caractère<CTRLH><CTRLN> et <CTRLN>caractère<CTRLO> de prefiltre.

SORTIE :

Un fichier (document formatè) où

- les commandes de NEQN et celles gènèrées par prefiltre ont ètè traitées afin d'obtenir une visualisation du document dans lequel les expressions mathématiques sont construites de manière aussi satisfaisante que possible sur le VT100, avec les commandes du VT100.

- les changements de polices de caractères et le passage aux caractères gras sont signalès grâce à l'utilisation d'attributs visuels.

- Certaines commandes provenant de la Sanders sont transposées en commandes du VT100 tandis que d'autres sont ignorées.

SPECIFICATIONS :

Ce programme effectue les traitements suivants :

- Il adapte à l'écran les caractères gènèrès par le deuxième jeu de caractères de neqn.Cette adaptation se fait de la manière suivante : certains caractères tels que π,... sont connus par le VT100,donc pas de problème. D'autres,comme la plupart des caractères grecs,n'existent pas sur le VT100.Sur l'écran,apparaîtront à la place,des caractères ascii normaux en noir sur blanc pour signifier que ce sont des caractères particuliers. D'autres,enfin,tels que l'intègrale,la somme,... sont construits de manière aussi proche de la rèalitè que possible.Il n'y a donc pas de problème pour les reconnaître.

- Les caractères spèciaux identifiès par prefiltre ne sont pas repròsentables à l'écran et sont remplacès par un point d'interrogation en noir sur blanc.

- Certains signes particuliers tels que $\geq, \leq, \neq$ et $\pm$ sont gènèrès par construction dans neqn.Ces caractères font partie des caractères graphiques de VT100.On peut donc les gènèrer à

l'écran.

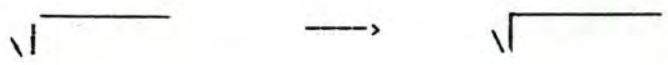
- Ce programme traite aussi,des commandes commençant par ‹ESC› :
  - certaines de ces commandes sont ignorées;
  - d'autres sont adaptées pour le VT100.

- Les backspaces posent aussi un problème,mais différent du cas de l'interface Sanders.Dans écran,il faut modifier la séquence de backspaces dans deux cas :

  - si sa présence est due à une barre de fraction,il la descendre afin qu'elle soit centrée par rapport au signe d'égalité.

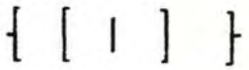
  - si sa présence est due à un soulignement,il faut éliminer les n back_spaces et les n "_" qui suivent et intercaler des commandes de début et de fin de soulignement autour des n caractères imprimables précédents (les commandes doivent être évitées).

- La racine carrée construite par neqn est remaniée afin qu'elle soit tracée de la manière la plus continue possible.



- Ecran remplace la suite discontinue de traits verticaux par des grands crochets,des grandes barres.Les grandes parenthèses sont similaires aux crochets et les accolades ont des coins carrés au lieu de coins arrondis.

NEQN :

ecran : 

- Ce programme utilise les attributs visuels du VT100 pour signaler un passage aux caractères gras ou une modification de police de caractères.

- Il reste un dernier problème à résoudre concernant les caractères "blancs".A l'écran,si on écrit un caractère blanc sur un caractère déjà écrit,ce dernier s'efface.Afin de ne pas perdre d'informations concernant les expressions mathématiques,on remplace les blancs par une commande de l'écran qui permet d'avancer le curseur d'un caractère.

3.4.2 La réalisation des traitements.

La découpe choisie pour réaliser ces traitements est illustrée à la figure 5.Cette structure est très analogue à celle

de back_space.
La différence essentielle réside dans le fait qu'à l'écran,deux caractères ne peuvent se superposer.Cette préoccupation engendre quelques petites modifications.
D'autre part,si l'écran offre moins de possibilités que l'imprimante quant aux caractères spéciaux,la largeur des caractères présents est identique et certains traitements tels que le traitement des backspaces,.. s'en trouvent facilités.

De même que pour back_space et prefiltre,les spécifications des modules intervenant dans la découpe ont été laissées en annexe.

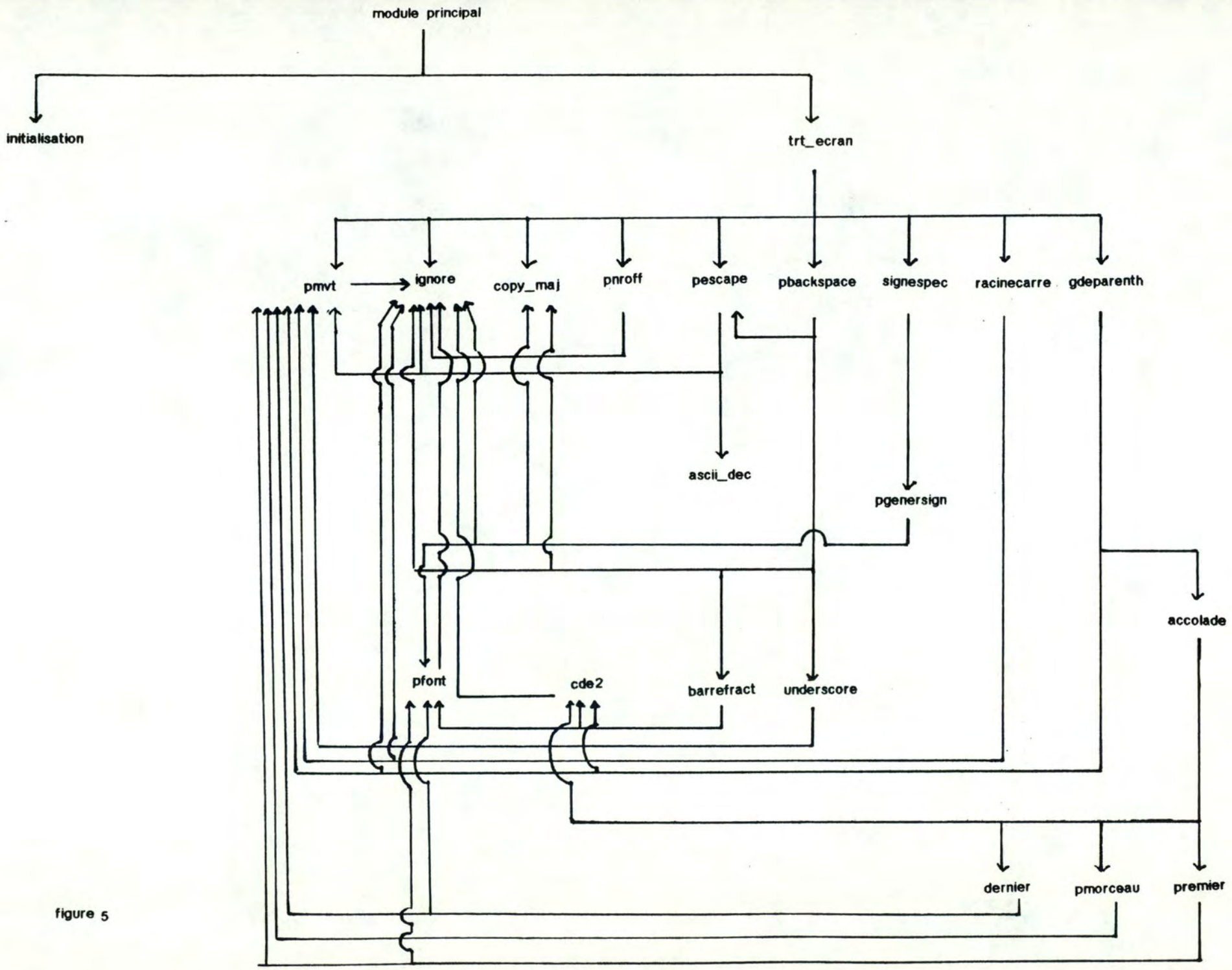

module principal
initialisation
trt_ecran
pmvt
ignore
copy_maj
pnroff
pescape
pbackspace
signespec
racinecarre
gdeparenth
ascii_dec
pgenersign
accolade
pfont
cde2
barrefract
underscore
dernier
pmorceau
premier

figure 5

Chapitre 5: DIFFICULTES,PROBLEMES.

Les deux chapitres précédents ont proposé une manière d'inclure des expressions mathématiques et des tableaux dans un document introduit sur le système UNIX des facultés.Le système élaboré possède ,néanmoins,quelques inconvénients.

Tout d'abord,le matériel choisi pour visualiser le document est un VT100.Vu ses propriétés non semi-graphiques,les expressions mathématiques,qui apparaissent à l'écran,ne sont pas identiques à celles qui sont imprimées.Les raisons en sont exposées au chapitre 3 (paragraphe 3.3).

L'utilisateur est loisible de changer de polices de caractères.Il doit néanmoins savoir que ce changement d'écriture s'effectuera au détriment de la justification à droite du texte.De plus,lorsque une ligne est écrite entièrement avec une police particulière,elle a une hauteur définie par le jeu de caractères.La distance entre deux lignes peut donc aussi varier.

Un troisième point à relever concerne le temps d'exécution de "back_space".Quoiqu'aucune mesure précise n'ait été effectuée,on peut constater que pour des fichiers d'un certain volume,le temps d'exécution est important,même lorsque le fichier de départ comporte peu d'équations mathématiques.Pour pallier cet état de fait, certaines possibilités sont proposées dans la conclusion de ce mémoire.

Le système offre,par l'intermédiaire de TBL,des possibilités d'élaboration de tableaux.Ceux-ci ne peuvent être construit qu'avec l'option "-ms" de NROFF et ne sont probablement pas aussi riches que l'utilisateur le désirerait : tout d'abord ces tableaux ne sont pas encadrés;de plus, on voudrait bien pouvoir introduire dans une colonne,une phrase entière qui serait formatée sur plusieurs lignes.Un tableau du type de la figure 1 reste un voeu non réalisé pour le moment.

New York Area Rocks

Era	Formation	Age (years)
Precambrian	Reading Prong	>1 billion
Paleozoic	Manhattan Prong	400 million
Mesozoic	Newark Basin, incl. Stockton, Lockatong, and Brunswick formations; also Watchungs and Palisades.	200 million
Cenozoic	Coastal Plain	On Long Island 30,000 years; Cretaceous sediments redeposited by recent glaciation.

figure 1

Reste un autre problème à signaler.Lors de la construction des expressions mathématiques,la consommation de mémoire est assez importante.La ligne spécifiant la formule peut atteindre l'ordre de 2000 caractères.Dans certains cas,l'équation dépasse la capacité de neqn et le système de formatage nous communique alors le petit message "word overflow".On peut néanmoins observer que,dans certains cas,si ce message apparaît avec l'option "-mc",il s'estompe avec l'option "-ms".

CONCLUSION

Dans ce travail,nous avons introduit le traitement de documents et tentè de donner un aperçu du marchè.Nous avons ensuite ètudiè de plus près les mèthodes d'èdition et de formatage.

Par la suite,nous nous sommes atttachès à un problème plus spècifique : l'adjonction de possibilitès mathèmatiques au système UNIX des facultès.Nous avons analysè son approche, prèsentè ses possibilitès et ses principes de rèalisation.

Nous avons ensuite proposè un procèdè qui permet aux profanes en mathèmatiques,mais toutefois initiès au maniement de base de l'ordinateur,de rèsoudre les problèmes qui pourraient leur être posès.

Nous n'avons pas manquè,pour terminer,de signaler quelques difficultès qui nous sont apparues au cours de nos recherches.En effet,pour obtenir un système de formatage et de visualisation sur UNIX pleinement satisfaisant, de nombreux aspects restent à exploiter et à dèvelopper.

Au chapitre 5,nous avons soulignè le temps d'exècution de "back_space". Les documents contiennnent,en gènèral,beaucoup de texte et peu d'èquations."back_space" n'est nècèssaire que dans la mesure où le texte introduit contient des èquations,des changements de polices ou des caractères gras.
Mise à part,une amèlioration des performances au niveau du programme lui-même (rèduction des ècritures,...),on peut envisager deux possibilitès:
Tout d'abord,on pourrait envisager la gestion d'un fichier de commandes.En effet,le passage dans "prefiltre" permet de dèterminer si le passage ultèrieur dans "back_space" s'avère utile et donc èviter une exècution sans intèrêt.
Ensuite,on pourrait concevoir une intègration de "back_space" avec le programme "vario" qui gère actuellement l'impression d'un document sur la Sanders.Le temps d'exècution de "back_space" ètant en partie dû à de nombreuses ècritures,perdrait un peu de son importance car l'impression sur papier de chaque ligne serait consècutif à son traitement.

Actuellement,il est possible de visualiser des documents contenant des expressions mathèmatiques sur VT100.Il faudrait assurer une compatibilitè au niveau des autres ècrans (Micro Bee ...).Le traitement d'un ècran à l'autre est analogue mais il faut,p.ex., construire un tableau contenant les commandes de dèplacements latèraux et verticaux ainsi que les commandes d'attributs visuels de chaque terminal.Les colonnes contiennent les commandes d'un terminal spècifique.Les lignes contiennent les commandes correspondantes pour chaque terminal.Il suffit donc,au dèpart,que le programme connaisse le terminal sur lequel on travaille pour sèlectionner la colonne adèquate.

Il faut souligner aussi les possibilitès importantes que peut offrir "prefiltre" quant à la manipulation du texte introduit.

Un outil d'aide devrait être dèveloppè afin que l'utilisateur connaisse les raisons des erreurs produites à l'exècution (dèpassement

de capacité,...) et qu'il puisse agir en conséquence.

Le système UNIX dispose de nombreux jeux de macros-instructions (-mc,-mvario,-ms,...).Chacun d'entre eux offre des possibilités intéressantes et différentes.Il serait utile de les rassembler et de les uniformiser afin de pouvoir les exploiter simultanément.

Comme cela a été souligné précédemment,les tableaux n'offrent pas tous les avantages que pourrait en attendre un utilisateur.A ce niveau aussi,des développements sont à étudier,notamment les possibilités d'encadrement et d'introduction dans une colonne,d'une phrase entière formatée sur plusieurs lignes ... Cette étude pourrait être basée sur un modèle de boîtes.

Les possibilités graphiques de la Sanders pourraient être exploitées dans d'autres domaines que les expressions mathématiques.Certaines figures simples telles que les encadrements pourraient être utiles à l'intérieur même d'un texte.

BIBLIOGRAPHIE

[ALLEN 81] ALLEN T.,NIX R.,PERLIS A.
"PEN : a hierarchical document editor"
Yale University - ARCH *1982*

[BMB 82] DAMBLON M.
"traitement de texte : panorama 82"
bmb - l'equipement de bureau et de l'informatique en Belgique,*mai - juin 82*

[BURGES 83] "Le traitement de texte"
Bureau Gestion -*83*

[CHAMBE 81] CHAMBERLIN D.D.,KING J.C.,SLUTZ D.R.,TODD S.J.P.,WADE W.
"JANUS : an interactive system for document composition"
IBM-ACM *1981*

[CROZET 80] CROZET J.M.,SERAIN D.
"Le langage PASCAL"
edition Masson,Paris New-York Barcelone Milan *1980*

[FURUTA 82] FURUTA R,SCOFIELD J.,SHAW A.
"Document Formatting Systems : Survey,Concepts and Issues"
Computing Surveys,Vol.14,no 3,*September 1982*

[GEFFROY 77] GEFFROY CH.
"Le traitement de texte"
La Revue de l'Entreprise,no 9,pp 30-34,*september 1977*

[HAMMER 81] HAMMER M.,ILSON R.,ANDERSON T.,GOOD M.,ROSENSTEIN L.,
NIAMIR B.,SCHOICHET S.,GILBERT E.
"The implementation of Etude,an integrated and interactive
document preparation system"
MIT-ACM *1981*

[KERN 75] KERNIGHAN,B.W.,CHERRY L.L.
"A system for typesetting mathematics"
Commun.ACM,Vol 18,no 3,pp151-157,*March 1975.*

[KERN] KERNIGHAN,B.W.
"Programming in C - A Tutorial"
Bell Laboratories,Murray Hill,N.J.

[KERN] KERNIGHAN,B.W.
"UNIX for beginners"
Bell Laboratories,Murray Hill,N.J.

[KERN 78] KERNIGHAN,B.W.,LESK M.E.,OSSANA J.F.
"UNIX time sharing system : Document preparation"
Bell system tecn. J.,Vol 57,no 6,pp 2115-2135,*aug.1978*

[KERN 81] KERNIGHAN,B.W.
"PIC - A crude graphics langague for typesetting"
Computer Science Tech. Rep.,Vol 85,*Jan. 1981*

BIBLIOGRAPHIE

[KNUTH 79] KNUTH,D.E.
"TEX and Metafont : New Directions in typesetting"
Digital Press and the American Mathematical Society,Bedford Mass.,and Providence,R.I.,*1979*

[LESK 76a] LESK M.E.
"TBL - A program to format tables"
Computer Science Tech.Rep. 49,Bell Lab.,Murray Hill,NJ,*Sept 1976*

[LESK 76b] LESK M.E.
"Typing documents on the UNIX System : using the -ms macros with TROFF and NROFF"
Internal Memo,Bell Laboratories,*October 1976*

[MICRO BEE] Technical user manuel - MICRO B
Video display terminals DM 20,MCB2

[OSSANA 74] OSSANA J.F.
"NROFF User's Manual"
Bell Laboratories,New Jersey,*nov. 74*

[QUINT 82] QUINT V.
"TITUS : un système pour l'édition interactive des formules mathématiques"
Labor. d'info. et de math. appl. de Grenoble RR no 306 ,*juin 1982*

[SANDERS] User's Guide For The Media 12/7
Typographic Printer
Sanders Technology

[SCHNE 78] SCHNEIDER G.M.,WEINGART S.W.,PERLMAN D.M.
"A introduction to programming and problem solving with PASCAL"
John Wiley & Sons New-York Chichester Brisbane Toronto ,*1978*

[VT100] DIGITAL EQUIPMENT CORPORATION
"User Guide for VT100"
january 1979

FACULTES UNIVERSITAIRES N.D. DE LA PAIX

NAMUR

INSTITUT D'INFORMATIQUE

DEVELOPPEMENT D'UN OUTIL DE TRAITEMENT MATHEMATIQUE SUR UNIX (ANNEXES)

PROMOTEUR :

PH. VAN BASTELAER

M.P. VANDEN BERGHE

ANNEE ACADEMIQUE : 1982.83

ANNEXE 1: QUELQUES DEFINITIONS

concatènation :

opèration qui consiste à assembler à la suite plusieurs èlèments afin de ne constituer qu'un seul ensemble.On parle de concatènation de caractères (pour former des chaînes),de concatènation de fichier.

courrier rèpètitif :

èlaboration rèpètitive de documents qui ne sont en fait que la repròsentation de textes prèexistants modifiès ou adaptès sur peu de points.L'exemple type en est la lettre circulaire reproduite à un grand nombre d'exemplaires "originaux" avec adjonction de quelques èlèments variables (nom du destinataires ...).

crayon lumineux :

dispositif d'interaction graphique en forme de crayon dont le fonctionnement est basè sur la dètection d'un signal lumineux et sa transformation en impulsion èlectrique.

edit et mince : noms d'èditeur

èditeur orientè-ècran :

est caractèrisè par le fait qu'il permet la manipulation d'un ècran considèrè comme un tout et qu'il se base essentiellement sur l'emploi d'un pointeur permettant à tout moment la rèfèrence à un point quelconque de cet ècran.

gènèrer : (mot anglais : generate) produire

implèmentation :

traduction directe du mot anglais "implementation",utilisèe en informatique pour dèsigner la rèalisation d'un logiciel,d'un progiciel ou d'un système d'exploitation et,par extension,sa mise en oeuvre.

identifiant :

nom symbolique donnè à un èlèment d'information qui permet de retrouver celle-ci sans ambiguitè.

interface :

jonction entre deux matèriels ou logiciels leur permettant d'èchanger des informations par l'adoption de règles communes physiques ou logiques.

imprimante caractère par caractère :

dèsigne un dispositif d'impression analogue à celui d'une machine à

écrire connectée.
Une boule ou quelques roues portant un seul jeu de caractères se déplacent le long de la ligne à imprimer.(30 à 60 caractères par seconde).

imprimante à jet d'encre :

imprimante à grande vitesse et excellente qualité d'impression dans laquelle les gouttelettes d'encre sont directement projetées sur un papier ordinaire après avoir traversé un champ électrique qui les charge électrostatiquement,puis sont disposées aux endroits voulus grâce à une grille directionnelle (45000 lignes/min. env.).

imprimante matricielle ou par point :

imprimante dans laquelle les caractères sont produits à partir d'une matrice d'aiguilles (généralement 5x7) : un caractère est dessiné par sélection de certaines aiguilles à l'aide d'une came;l'encrage préalable des aiguilles permet alors d'obtenir la trace correspondante sur le papier. (env. 200 c/sec).

imprimante à laser :

imprimante telle que chaque ligne est composée de points électrisés qui se chargent de poudre de carbone,fixée ensuite à la chaleur.(Une telle imprimante édite 10000 pages à l'heure).

macro-instruction :

procédé d'écriture permettant à un programmeur de définir des opérations composées à partir des instructions du répertoire de base d'un système donné.
Les macros-instructions permettent de rendre l'écriture d'un programme plus concise en évitant la répétition de séquences d'instructions semblables.

marguerite :

roue de plastique ou de métal,située dans l'imprimante,assurant l'impression des caractères sur le papier.Sa forme ronde lui permet d' imprimer jusqu'à 540 mots à la minute.

menu :

liste de commandes ou options offertes au choix d'un utilisateur lors d'un travail en mode conversationnel.

Le menu est particulièrement utilisé en Infographie interactive, par affichage sur une partie réservée de l'écran.Les choix de l'utilisateur se font alors soit à l'aide d'un dispositif de pointage, soit en déplaçant une souris,soit soit par l'entrée de données à l'aide d'un clavier.Chaque commande d'un menu peut donc faire apparaître un nouveau menu et ainsi de suite.La commande ou option choisie est désignée à l'utilisateur par le système,par surbrillance,clignotement,... pour confirmation de l'option prise.

module

sous-ensemble d'un programme destinè à rèaliser un ensemble de fonctions prècises.
L'utilisation de modules permet de dècomposer un programmes en parties indèpendantes,communiquant entre elles à l'aide d'interfaces spècifiès de façon très prècise.

police :

(mot anglais : font) ensemble de caractères formant un jeu.

progiciel :

programmes standards conçus pour une application commune à plusieurs utilisateurs,soit tels quels,soit sous rèserve d'adaptations mineures.

souris :

boule èlectronique de la taille de la paume de la main que l'on dèplace sur la table de travail et qui commande un curseur sur l'écran.

tilde : "~"

ANNEXE 2: TESTS

Cette annexe regroupe quelques exemples et montre les possibilités du système proposé.
Pour la plupart d'entre eux,on donne le texte obtenu et le texte tel qu'il est introduit afin d'avoir un point de comparaison.

Tout d'abord,on trouve un texte formaté avec nroff -mc et nroff -ms; celui-ci est suivi des deux textes sources.Ensuite,quelques possibilités de nroff -ms sont présentées (entêtes,double colonne,...). Ceux-ci sont suivis des équations mathématiques et des tableaux.

EXEMPLE

1. ceci est un exemple de texte _formate_ a l'aide des commandes de mise en page decrites dans le memorandum IM/03.

Afin de bien saisir le role de ces commandes, il importe de comparer le texte tel qu'il est introduit dans l'ordinateur avec le resultat produit par l'application du programme de mise en page ("nroff").

Un certain nombre de regles simples doivent etre respectees pour permettre une utilisation efficace de "nroff":

1. Chaque commande de mise en page commence obligatoirement au debut d'une ligne et se compose d'un point (".") suivi du nom de la commande et, parfois, d'un nombre ou d'une lettre.

2. le programme de mise en page compose automatiquement des lignes de 80 caracteres, sans se preoccuper de la disposition du texte original. Ce sont les commandes

```
.PP    (nouveau paragraphe)
.S N   (passer a la ligne en passant N lignes blanches)
```

qui permettent de produire des lignes incompletes.

3. les commandes .DS et .DE entourent un texte que l'on desire faire passer dans le resultat sans transformation (si ce n'est un eventuel decalage vers la droite).

D'autres commandes interessantes sont:

- production de paragraphes precedes d'un tiret: .TP et .RP;
- production de sections numerotees automatiquement : .NH et .NP;
- production de paragraphes numerotes automatiquement: .NO et .NE;

Un exemple de section numerotee automatiquement:

1.1. tralala

Et voici le texte ...

1.2. Youplaboum

Et voici encore du texte ...

2. Retour au premier niveau de section

il est parfois interessant de retrecir un morceau de texte pour en accentuer la portee; cela se fait, comme dans cet exemple-ci, a l'aide des commandes .SQ et .SE.

EXEMPLE

2.1. Ceci montre que les numeros de section sont incrementes automatiquement.

Et voici un exemple de
texte
centre
automatiquement.
pratique, non?

On peut egalement, comme le montre cette partie de texte, faire en sorte que la marge de gauche soit decalee de 5 positions vers la droite Il faut,pour cela,utiliser la commande ,.RS avant le debut du texte a decaler , et la commande .RE a la fin de ce texte.

Et nous voici revenus a la position normale de la marge de gauche; cette pos position tient compte, bien entendu, du niveau de section dans laquelle on se trouve.

```
.TL "" "EXEMPLE" ""
.TR
.NP 1
ceci est un exemple de texte
.UL 1
formate
a l'aide des commandes de mise en page decrites dans le memorandum IM/03.
.S 1
Afin de bien saisir le role de ces commandes, il importe de comparer
le texte tel qu'il est introduit dans l'ordinateur avec le resultat
produit par l'application du programme de mise en page ("nroff").
.PP 2
Un certain nombre de regles simples doivent etre respectees pour permettre
une utilisation efficace de "nroff":
.S1
.NO
Chaque commande de mise en page commence obligatoirement au debut d'une ligne
et se compose d'un point (".") suivi du nom de la commande et, parfois, d'un
nombre ou d'une lettre.
.S 1
.NO
le programme de mise en page compose automatiquement des lignes de 80 caracteres,
sans se preoccuper de la disposition du texte original. Ce sont les commandes
.DS I
 .PP    (nouveau paragraphe)
 .S N   (passer a la ligne en passant N lignes blanches)
.DE
qui permettent de produire des lignes incompletes.
.S 1
.NO
les commandes .DS et .DE entourent un texte que l'on desire faire passer dans
le resultat
.UL 1
sans transformation
(si ce n'est un eventuel decalage vers la droite).
.S1
.NE
.PP2
D'autres commandes interessantes sont:
.S 1
.TP
production de paragraphes precedes d'un tiret: .TP et .RP;
.S 1
.TP
production de sections numerotees automatiquement : .NH et .NP;
.S 1
.TP
production de paragraphes numerotes automatiquement: .NO et .NE;
.RP
.S 2
Un exemple de section numerotee automatiquement:
.NH 2
tralala
.PP 1
Et voici le texte ...
.NH 2
Youplaboum
.PP 1
Et voici encore du texte ...
.NH 1
Retour au premier niveau de section
.SQ
il est parfois interessant de retrecir un morceau de texte pour en accentuer
la portee; cela se fait, comme dans cet exemple-ci, a l'aide des commandes .SQ
et .SE.
.SE
```

```
.NP 2
Ceci montre que les numeros de section sont incrementes automatiquement.
.DS C
Et voici un exemple de
texte
centre
automatiquement.
pratique, non?
.DE
.RS
.PP 1
On peut egalement, comme le montre cette partie de texte, faire en sorte que
la marge de gauche soit decalee de 5 positions vers la droite
Il faut,pour cela,utiliser la commande ,.RS avant le debut du texte a decaler
, et la commande .RE a la fin de ce texte.
.RE
.PP 2
Et nous voici revenus a la position normale de la marge de gauche; cette pos
position tient compte, bien entendu, du niveau de section dans laquelle
on se trouve.
```

```
.EG
.ND
.TL
"EXEMPLE"
.NH 1
.PP

ceci est un exemple de texte
.UL 1
formate
a l'aide des commandes de mise en page decrites dans le memorandum IM/03.
.sp 1
Afin de bien saisir le role de ces commandes, il importe de comparer
le texte tel qu'il est introduit dans l'ordinateur avec le resultat
produit par l'application du programme de mise en page ("nroff").
.PP 2
Un certain nombre de regles simples doivent etre respectees pour permettre
une utilisation efficace de "nroff":
.sp 1
.IP 1.
Chaque commande de mise en page commence obligatoirement au debut d'une ligne
et se compose d'un point (".") suivi du nom de la commande et, parfois, d'un
nombre ou d'une lettre.
.sp 1
.IP 2.
le programme de mise en page compose automatiquement des lignes de 80 caracteres,
sans se preoccuper de la disposition du texte original. Ce sont les commandes
.DS I
 .PP    (nouveau paragraphe)
 .S N   (passer a la ligne en passant N lignes blanches)
.DE
qui permettent de produire des lignes incompletes.
.sp 1
.IP 3.
les commandes .DS et .DE entourent un texte que l'on desire faire passer dans
le resultat
.UL 1
sans transformation
(si ce n'est un eventuel decalage vers la droite).
.sp 1
.LP
.PP2
D'autres commandes interessantes sont:
.IP -
production de paragraphes precedes d'un tiret: .TP et .RP;
.IP -
production de sections numerotees automatiquement : .NH et .NP;
.IP -
production de paragraphes numerotes automatiquement: .NO et .NE;
.LP
.sp 2
Un exemple de section numerotee automatiquement:
.NH 2
tralala
.PP 1
Et voici le texte ...
.NH 2
Youplaboum
.PP 1
Et voici encore du texte ...
.NH 1
Retour au premier niveau de section
.PP

.RS
il est parfois interessant de retrecir un morceau de texte pour en accentuer
```

```
la portee; cela se fait, comme dans cet exemple-ci, a l'aide des commandes .SQ et .SE.
.RE
.IP 2.1
Ceci montre que les numeros de section sont incrementes automatiquement.
.LP
.DS C
Et voici un exemple de
texte
centre
automatiquement.
pratique, non?
.DE
.RS
.PP 1
On peut egalement, comme le montre cette partie de texte, faire en sorte que
la marge de gauche soit decalee de 5 positions vers la droite
Il faut,pour cela,utiliser la commande ,.RS avant le debut du texte a decaler
, et la commande .RE a la fin de ce texte.
.RE
.PP 2
Et nous voici revenus a la position normale de la marge de gauche; cette pos
position tient compte, bien entendu, du niveau de section dans laquelle
on se trouve.
```

The Role of Allen Wrench in Modern Electronics

J.Q. Pencilpusher

X.Y.Hardwired

Bell Laboratories
Murray Hill,New Jersey 07974

ABSTRACT

This abstract should be short enough to fit on a single page cover sheet. It must attract the reader into sending for the complete memorandum.

The Role of Allen Wrench in Modern Electronics

J.Q. Pencilpusher

X.Y.Hardwired

Bell Laboratories
Murray Hill,New Jersey 07974

1. Introduction

The solution to the torque handle equation

$$\sum_{0}^{\infty} F(x_i) = G(x) \tag{1}$$

is found with the transformation $x=\frac{\rho}{\theta}$ where $\rho=G'(x)$ and θ is derived from well-know principles.

A SIMPLE LIST

1. J.Pencilpusher and X. Hardired. *A New Kind of Set Screw.* Proc. IEEE 75 (1976),23-255

2. H. Nails and R. Irons. *Fasteners for Printed Circuit Boards,* Proc. ASME 23 (1974),23-24

The Declaration of Independance

When in the course of human events,it becomes necessary for one people to dissolve the political bonds which have connected them with another,and to assume among the powers of the earth the separate and equal station to which the laws of Natur's God entitle them,a decent respect to opinions of mandkind requires that they should declare the causes which impel them to the separation.

We hold these truths to be self-evident,that all men are created equal,that are endowed by their creator with certain unalienable rights,that among these life,liberty,and the pursuit of happiness.That to secure these rights, governments are instituted among men,...

Multiple Indents

This is ordinary text to point out

the margins of the page.

1. First level item

   a) Second Level.

   b) Continued here with another second level item,but somewhat longer.

2. Return to precious value of the identing at this point.

3. Another line.

```
.EQ
delim $$
.EN
.RP
.ND April 1,1976
.TL
The Role of Allen Wrench in Modern Electronics
.AU
J.Q. Pencilpusher
.AU
X.Y.Hardwired
.AI
Bell Laboratories
Murray Hill,New Jersey 07974
.AB
This abstract   should be short enough to fit on a single page cover sheet.
It must attract the reader into sending for the complete memorandum.
.AE
.NH
Introduction
.PP
The solution to the torque handle equation
.EQ (1)
sum from 0 to inf F ( x sub i ) = G (x)
.EN
is found with the transformation $x = rho over theta $ where $ rho = G ' (x) $
and $theta$ is derived from well-know principles.
.LP
.sp 3
.BO
A SIMPLE LIST
.sp
.IP 1.
J.Pencilpusher and X. Hardired.
.I
A New Kind of Set Screw.
Proc. IEEE
.BO
75
(1976),23-255
.IP 2.
H. Nails and R. Irons.
.I
Fasteners for Printed Circuit Boards,
Proc. ASME
.BO
23
(1974),23-24
.LP
.2C
.SH
The Declaration of Independance
.PP
When in the course of human events,it becomes necessary for one people
to dissolve the political bonds which have connected them with another,and
to assume among the powers of the earth the separate and equal station
to which the laws of Natur's God entitle them,a decent respect to opinions
of mandkind requires that they should declare the causes which impel
them to the separation.
.PP
We hold these truths to be self-evident,that all men are created equal,that
are endowed by their creator with certain unalienable rights,that among these
life,liberty,and the pursuit of happiness.That to secure these rights,
governments are instituted among men,...
.sp 2
.BO
```

```
Multiple Indents
.sp 2
This is ordinary text to point out the margins of the page.
.IP 1.
First level item
.RS
.IP a)
Second Level.
.IP b)
Continued here with another second level item,but somewhat longer.
.RE
.IP 2.
Return to precious value of the identing at this point.
.IP 3.
Another
line.
```

EXEMPLES

$$x=2\pi\int\sin(\omega t)dt$$

$$e^{i\omega t}$$

$$x_{i_1}^2$$

$$e^{i\pi^{\rho+1}}$$

$$\frac{a+b}{2c}=1$$

$$\frac{a+b}{c+d+e}=1$$

$$\frac{\partial^2 f}{\partial x^2}=\frac{x^2}{a^2}+\frac{y^2}{b^2}$$

$$\frac{\alpha+\beta}{\sin(x)}$$

$$\frac{-b^2}{\pi}$$

$$\sqrt{a+b}+\frac{1}{\sqrt{ax^2+bx+c}}$$

$$\sqrt{\frac{a^2}{b_2}}$$

$$x=\frac{-b\pm\sqrt{b^2-4ac}}{2a}$$

$$\frac{x^2}{a^2}=\sqrt{pz^2+qz+r}$$

$$erf(z)=\frac{2}{\sqrt{\pi}}\int_0^z e^{-t^2}dt$$

$$x=\frac{-b\pm\sqrt{b^2-4\sqrt{ac}}}{\sqrt{2a}}$$

$$\sum_{i=0}^{\infty} x_i=\frac{\pi}{2}$$

EXEMPLES

$$\sum_{i=0}^{i=\infty} x^i$$

$$\zeta(s) = \sum_{k=1}^{\infty} k^{-s} \quad (Res>1)$$

$$\lim_{n\to\infty} x^n = 0$$

$$\lim_{x\to\pi/2} (\tan x) = \infty$$

$$\lim_{x\to\pi/2} (\tan x)^{\sin x} = 1$$

$$\prod_{i=0}^{n} a_i = 1$$

$$\bigcup_{i=0}^{n} a_i = 1$$

$$\mathop{\mathbf{A}}_{i=0}^{n} a_i = 1$$

$$\left[\frac{x+y}{2a}\right] = 1$$

$$\left\{\frac{a}{b+1}\right\} = \left[\frac{c}{d}\right] + [\,e]$$

$$A = \begin{bmatrix} a & x \\ b & y \\ c & z \end{bmatrix}$$

EXEMPLES

$$\text{sign}(x) \equiv \begin{cases} 1 & \text{if } x>0 \\ 0 & \text{if } x=0 \\ -1 & \text{if } x<0 \end{cases}$$

$$\begin{matrix} a & c & e \\ b & d & f \end{matrix}$$

$$\begin{vmatrix} x_i & x^2 \\ y_i & y^2 \end{vmatrix}$$

$$\dot{x}+\ddot{X}+\dot{y}+\dot{y} + \overline{x+y} + \overline{\alpha+\beta}+x$$

Let α_i be the primary variable, and let β be zero. Then we can show that x_1 is ≥ 0 .

$$x+y=z$$

$$x=1$$

$$\lim_{k\to\infty} \min_{x' \in A} \rho_1(k)\, \rho_{\rho_1(k)}(x') = 0$$

```
.TL "" "EXEMPLES" ""
.TR
.S 1
.EQ
x = 2 pi int sin ( omega t)dt
.EN
.S 1
.EQ
e sup {i omega t}
.EN
.S 1
.EQ
x sub {i sub 1} sup 2
.EN
.S 1
.EQ
e sup {i pi sup {rho + 1}}
.EN
.S1
.EQ
a+b over 2c =1
.EN
.S 1
.EQ
a+b over c+d+e =1
.EN
.S 1
.EQ
{partial sup 2 f} over {partial x sup 2} =
x sup 2 over a sup 2 + y sup 2 over b sup 2
.EN
.S 2
.EQ
{alpha + beta} over {sin(x)}
.EN
.S 1
.EQ
-b sup 2 over pi
.EN
.S 1
.EQ
sqrt a+b ~+~ 1 over sqrt {ax sup 2 +bx+c}
.EN
.S 3
.EQ
sqrt {a sup 2 over b sub 2}
.EN
.S 2
.EQ
x = {-b +- sqrt{b sup 2 -4ac}} over 2a
.EN
.S 1
.EQ
x sup 2 over a sup 2 ~=~ sqrt {pz sup 2 + qz + r}
.EN
.S 1
.EQ
erf (z) ~=~ 2 over sqrt pi int sub 0 sup z e sup -t sup 2 dt
.EN
.S 1
.EQ
x= {-b +- sqrt {b sup 2 -4 sqrt {ac}}} over sqrt {2a}
.EN
.S 3
.EQ
sum from i=0 to infinity x sub i = pi over 2
```

```
.EN
.S 1
.EQ
sum from i=0 to {i= inf} x sup i
.EN
.S 3
.EQ
zeta (s) ~=~ sum from k=1 to inf
k sup -s  ~~~  (Re s > 1)
.EN
.S 3
.EQ
lim from {n -> inf} x sup n =0
.EN
.S 3
.EQ
lim from {x -> pi /2} ( tan~x) = inf
.EN
.S 3
.EQ
lim from {x -> pi / 2} (tan ~x) sup {sin ~ x} ~ =~ 1
.EN
.S 1
.EQ
prod from i=0 to n a sub i =1
.EN
.S 3
.EQ
union from i=0 to n a sub i =1
.EN
.S 3
.EQ
inter from i=0 to n a sub i =1
.EN
.S 3
.EQ
left [ x+y over 2a right ] ~=~ 1
.EN
.S 3
.EQ
left { a over b+1 right } = left ( C over d right ) + left [ e right ]
.EN
.S 3
.EQ
A ~=~ left [
pile { a above b above c}~~pile {x above y above z}
right  ]
.EN
.S 3
.EQ
sign(x) ~==~ left {
   rpile {1 above 0 above -1}
  ~~lpile {if above if above if}
  ~~lpile {x>0 above x=0 above x<0}
.EN
.S 3
.EQ
matrix { lcol { a above b} ccol {c above d} rcol {e above f}}
.EN
.S 3
.EQ
left |
matrix {
  ccol {x sub i above y sub i}
  ccol {x sup 2 above y sup 2}
}
```

```
right |
.EN
.S 6
.EQ
x dot + X dot + y hat + y dotdot + x+y bar + {alpha + beta} bar + x tilde
.EN
.S 3
.EQ
delim $$
.EN
Let $alpha sub i$ be the primary variable,and let $beta$ be zero.Then we
can show that $x sub 1$ is $>=0$ .
.EQ
delim off
.EN
.S 3
.EQ
x+y mark = z
.EN
.S1
.EQ
x lineup = 1
.EN
.S 3
.EQ
lim from {k -> inf}~~ min from {x' app~A} phi sub 1 (k)~rho sub {phi sub 1 (k)}
(x')~=~0
.EN
.S 3
```

$$\mathrm{var}_{\mu,\nu}\left[\nu_T^n\right] = 2\nu^2 \cdot \left[1 - \sum_{i=1}^{n}\left[\frac{t_i^n - t_{i-1}^n}{T}\right]^2\right]^2$$

$$\frac{1}{2\pi}\int_0^{\sqrt{y}}\left[\sum_{k=1}^{n}\sin^2 x_k(t)\right]f(t)dt$$

```
.EQ
var sub {mu , nu}
left [ nu sub T sup n right ] ~=~
2 nu sup 2 .  left [
{1~-~ sum from i=1 to n
left [
{{ t sub i sup n - t sub {i-1} sup n}
over T  }
right ]
sup 2}
right ] sup 2
.EN
.sp 5
.EQ
1 over {2 pi} int sub 0 sup  sqrt {y}
left [
sum from k=1 to n sin sup 2 x sub k (t)
right ]
f(t) dt
.EN
```

$$\frac{x^2}{a^2} = \sqrt{pz^2+qz+r}$$

(1.3)

$$F(X) = |\nabla v|^2$$
$$= \left[\frac{\partial v}{\partial x}\right]^2 + \left[\frac{\partial v}{\partial y}\right]^2 \qquad \lambda \to \infty$$

```
.EQ (1.3)
x sup 2 over a sup 2 ~=~ sqrt {pz sup 2 + qz + r}
.EN
.sp
.sp
.EQ L
F hat ( chi ) ~ mark = ~ | del V | sup 2
.EN
.EQ L
lineup = ~ { left ( {partial V} over {partial x } right ) } sup 2~+~
{left ( {partial V} over {partial y} right ) } sup 2 ~~~~~~lambda -> inf
.EN
```

	titre
nom1	valeur1
nom2	valeur2
nom3	valeur3
nom4	valeur4

	titre	
nom1	valeur1	total1
nom2	valeur2	total2
nom3	valeur3	total3
nom4	valeur4	total4

	titre	
nom1	valeur1	total1
nom2	valeur2	total2
nom3	valeur3	total3
nom4	valeur4	total4

	Household Population	
Town	Households	
	Number	Size
Bedminster	789	3.26
Bernards Twp.	3087	3.74
Bernardsville	2018	3.30
Bound Brook	3425	3.04
Branchburg	1644	3.49
Bridgewater	7897	3.81
Far Hills	240	3.19

	Household Population	
Town	Households	
	Number	Size
Bedminster	789	3.26
Bernards Twp.	3087	3.74
Bernardsville	2018	3.3
Bound Brook	3425	3.104
Branchburg	1644	3.49
Bridgewater	7897	3.81
Far Hills	240	3.19

```
notation : \t = tabulation

.TS
cl sr
titre
nom1\tvaleur1
nom2\tvaleur2
nom3\tvaleur3
nom4\tvaleur4
.TE

.TS
cl sr sr
titre
nom1\tvaleur1\ttotal1
nom2\tvaleur2\ttotal2
nom3\tvaleur3\ttotal3
nom4\tvaleur4\ttotal4
.TE

.TS
cl8 sr8 sr
titre
nom1\tvaleur1\ttotal1
nom2\tvaleur2\ttotal2
nom3\tvaleur3\ttotal3
nom4\tvaleur4\ttotal4
.TE

.TS
cccl sccr sscr
Household Population
Town\tHouseholds
\tNumber\tSize
Bedminster\t789\t3.26
Bernards Twp.\t3087\t3.74
Bernardsville\t2018\t3.30
Bound Brook\t3425\t3.04
Branchburg\t1644\t3.49
Bridgewater\t7897\t3.81
Far Hills\t240\t3.19
.TE

.TS
cccl sccn sscn
Household Population
Town\tHouseholds
\tNumber\tSize
Bedminster\t789\t3.26
Bernards Twp.\t3087\t3.74
Bernardsville\t2018\t 3.3
Bound Brook\t3425\t3.104
Branchburg\t1644\t3.49
Bridgewater\t7897\t3.81
Far Hills\t240\t3.19
.TE
```

Major New York Bridges

Bridges	Designer	Length
Brooklyn	J.A.Roebling	1595
Manhattan	G. Lindenthal	1470
Williamsburg	L.L.Burck	1600
Queensborough	Palmer &Hornbostel	1182
Trigorought	O.H.Ammann	1380
		1383
Bronx Whitestone	O.H.Ammann	2300
Throgs Neck	O.H.Hamman	1800
George Washington	O.H.Ammann	3500

notation : \t = notation

```
.TS
ccl scl scn
Major New York Bridges
Bridges\tDesigner\tLength
Brooklyn\tJ.A.Roebling\t1595
Manhattan\tG. Lindenthal\t1470
Williamsburg\tL.L.Burck\t1600
Queensborough\tPalmer &Hornbostel\t1182
Trigorought\tO.H.Ammann\t1380
\t\t1383
Bronx Whitestone\tO.H.Ammann\t2300
Throgs Neck\tO.H.Hamman\t1800
George Washington\tO.H.Ammann\t3500
.TE
```

Name	Definition
Gamma	$\Gamma(z)=\int_0^\infty t^{z-1}e^{-t}dt$
Sine	$\sin(x)=\frac{1}{2i}(e^{ix}-e^{-ix})$
Error	$\operatorname{erf}(z)=\frac{2}{\sqrt{\pi}}\int_0^z e^{-t^2}dt$
Bessel	$J_0(z)=\frac{1}{\pi}\cos(z\sin\theta)d\theta$
Zeta	$\zeta(s)=\sum_{k=1}^{\infty} k^{-s}$ (Re s>1

Composition of Foods

Food	Percent by Weight		
	Protein	Fat	Carbohydrate
Apples	.4	.5	13.0
Halibut	18.4	5.2	...
Lima beans	7.5	.8	22.0
Milk	3.3	4.0	5.0
Mushrooms	3.5	.4	6.0
Rye bread	9.	.6	52.7

```
.EQ
delim $$
.EN


.TS
cl cl
Name\tDefinition
.sp

Gamma\t$GAMMA (z) = int sub 0 sup inf t sup {z-1} e sup -t dt $


Sine\t$sin (x) = 1 over 2i (e sup ix - e sup -ix)$


Error\t$erf(z) = 2 over sqrt pi int sub 0 sup z e sup {-t sup 2} dt $


Bessel\t$J sub 0 (z) = 1 over  pi cos ( z sin theta ) d theta $


Zeta\t$zeta (s) = sum from k=1 to inf k sup -s ~~ (Re~s > 1$
.TE
.sp
.sp
.TS
cccl sccn sscn sscn
Composition of Foods
Food\tPercent by Weight
\tProtein\tFat\tCarbohydrate
Apples\t.4\t.5\t13.0
Halibut\t18.4\t5.2\t...
Lima beans\t7.5\t.8\t22.0
Milk\t3.3\t4.0\t5.0
Mushrooms\t3.5\t.4\t6.0
Rye bread\t9.\t.6\t52.7
.TE
```

ANNEXE 3: MODES D'EMPLOI

Cette annexe offre un manuel d'utilisation à toute personne souhaitant formater un document sur UNIX.Elle se décompose en plusieurs parties :

1. Tout d'abord,on trouvera le jeu de macros-instructions "-mc" de nroff.

2. Ensuite,le jeu "-ms" avec ses différences par rapport à "-mc".

3. La troisième partie regroupe les macros-instructions qui permettent de changer de polices de caractères et de passer aux caractères gras.Ces instructions sont valables quel que soit le jeu de macros-instructions choisi.

4. La quatrième partie explique le langage d'introduction des équations mathématiques.

5. On clôture,finalement,par les spécifications nécessaires à la construction de tableaux.

Une fois le texte introduit,la suite la plus générale des commandes qui permettent l'impression du document formaté sur la Sanders est :

```
eml prefiltre.x texte.txt texte.pre                          (1)
tbl texte.pre | neqn | nroff (option1) > texte.vario         (2)
eml back_space.x texte.vario texte.bs                        (3)
vario (option2) texte.bs                                     (4)
```

où (option1) est -mc,-mvario ou -ms;
L'option "-ms" est néanmoins indispensable pour la création de tableau.
(option2) est facultative et peut valoir
-o : sortir sur un fichier au lieu de l'imprimante;
-d : imprimer en draft;
-h : help;
-sN : sélectionner la taille du papier;
-fN : sélectionner le style d'impression;
-uC : sélectionner la distance de soulignement C

Les commandes (1) et (3) sont inutiles si le texte ne contient que des instructions de la partie 1 ou 2 du manuel d'utilisation.La commande (3) se réduit à

```
neqn texte.pre | nroff (option1) > texte.vario               (5)
```

s'il n'y a pas de tableaux à construire et à

```
nroff (option1) texte.pre > texte.vario                      (6)
```

s'il n'y a pas d'équations mathématiques.

Afin de visualiser le document à l'écran,la commande à taper est :

eml ecran.x texte.vario texte.ecran (7)

Et pour revoir le texte,à nouveau:

cat texte.ecran (8)

PARTIE 1: LES MACROS-INSTRUCTIONS -MC

1. INTRODUCTION

Ces notes décrivent les macros définies pour les utilisateurs de NROFF qui désirent écrire des notes de cours ou des rapports.

Les macros mentionnées se trouvent dans /usr/lib/tmac.c.Il faudra donc lors du formatage utiliser l'option "-mc" pour que ces instructions soient prises en compte (nroff -mc file).

2. LES COMMANDES DE BASE DE FORMATAGE

Voici tout d'abord les différentes possibilités offertes.Une description plus complète sera donnée à la section suivante.

2.1. création de titres (pour les sections)

Tous ces titres sont soulignés.
- intitulé non numéroté : .SH
- intitulé numéroté selon son niveau de section : .NH N
- intitulé numéroté selon son niveau de section et centré : .CH N

2.2. création de paragraphes

- création d'un nouveau paragraphe avec alinéa à la première ligne : .PP N
  [cas particulier : .P1]
- paragraphe numéroté : .NP N
- paragraphe précédé d'un tiret : .TP
  [pour terminer l'énumération de paragraphes précédés d'un tiret : .RP]
- paragraphe précédé d'un numéro : .NO
  [pour terminer l'énumération de paragraphes précédés d'un numéro : .NE]

2.3. Modifications des marges,des alinéas.

- remise à jour du niveau courant de l'alinéa : .RL
- décalage de la marge de gauche

  .RS : augmentation
  .RE : diminution

2.4. traitements particuliers de parties de texte

- contraction de texte

  début de contraction : .SQ
  fin de contraction : .SE

- transcription littérale de texte

  début de transcription : .LS
  fin de transcription : .LE

- mise en évidence de lignes

  début de traitement : .DS
  fin de traitement : .DE

- insertion de figure

  début d'insertion : .SF
  fin d'insertion : .EF

- insertion de dessin : .PI N

2.5. Comment passer des lignes

.S N [cas particuliers : .S0 et .S1]

2.6. souligner des lignes

.UL N

2.7. Pour passer a la page suivante

.BP

2.8. Numérotation de pages

.PN N

2.9. Présentation générale : livre ou article technique

- définition du titre : .TL string-lh string-ch string-rh
- structure générale d'un livre :

  tête de chapitre : .CP N string (précédé par .TL)
  fin de chapitre : .CE
  option "chapitre","annexe",... : .ds TT string
  (1ère commande dans le texte)

- structure générale d'un article

Présentation du titre : .TR (précède par .TL)

- notes infra-paginales

  début de la note : .FS
  fin de la note : .FE

3. DESCRIPTION DES COMMANDES DE FORMATAGE

3.1. Création de titres (pour les sections)

Il y a différentes façons de créer un titre de section selon l'option choisie : l'intitulé peut être numéroté ou non,il peut être centré ou non.

3.1.1. .SH - Unnumbered Section Heading

Pour obtenir un titre non numéroté,tapez :

".SH "

et indiquez à la ligne suivante votre intitulé.Il sera automatiquement souligné.

3.1.2. .NH N - Numbered Section Heading

Pour obtenir un titre numéroté,tapez

" .NH N "

où N représente le numéro de niveau de la section.
ex: .NH 3 changera 2.1.2 en 2.1.3.
Si N est omis,le niveau est 1 par défaut.
A la ligne suivante,indiquez votre titre.Il sera souligné automatiquement ainsi que le numéro de section.La marge droite sera automatiquement augmentée ou diminuée en rapport avec le niveau de la section.

3.1.3. .CH N - Centered Section Heading

Pour obtenir un titre numéroté et centré,tapez :

" .CH N "

où N représente le numéro de niveau de la section.
Indiquez votre titre à la ligne suivante.
L'effet est analogue à celui de la commande précédente.Il y aura néanmoins plus d'espace avant l'intitulé qui sera centré.

3.1.4. Remarque

Pour chacune de ces instructions,le formateur considérera comme titre, tout le texte introduit jusqu'à la prochaine commande.C'est ce texte-là qui sera souligné.

Une commande ".NH 0" ou ".CH 0" remet la numérotation du niveau 1 à un.

3.1.5. exemple

Si vous tapez :

```
.SH
titre 1
.SH
titre 2
.SH
titre 3
```

vous obtenez

titre 1

titre 2

titre 3

Si vous tapez :

```
.NH
titre 1
.NH 2
titre 2
.NH 2
titre 3
.NH 3
titre 4
.CH 1
titre 5
.CH 2
titre 6
```

vous obtenez :

1. titre 1

1.1. titre 2

1.2. titre 3

1.2.1. titre 4

2. titre 5

2.1. titre 6

Renumérotation du 1er niveau à partir de 1 : tapez :

```
.NH 0
titre 7
```

Cela donne :

1. titre 7

3.2. Création de paragraphes

3.2.1. .PP N - New Paragraph

Pour créer un nouveau paragraphe,tapez :

" .PP N "

où N est le nombre de lignes à passer avant le début de celui-ci. A la première ligne de ce paragraphe,vous aurez un alinéa.

exemple : Si vous tapez :

```
.PP 2
Vous observez qu'on passe 2 lignes avant de commencer ce
nouveau paragraphe. Vous voyez aussi que la première
ligne est décalée.
```

vous obtenez :

Vous observez qu'on passe 2 lignes avant de commencer ce nouveau paragraphe. Vous voyez aussi que la première ligne est décalée.

Cas particuliers :

- Si vous n'écrivez que ".PP","N" aura,par défaut,la valeur 0.
- ".P1" équivaut à ".PP 1"

3.2.2. .NP N - Numbered Paragraph

Pour numéroter un paragraphe,tapez

" .NP N "

où N est le numéro du niveau.
Le numéro du paragraphe n'est pas souligné et la marge de gauche est automatiquement augmentée ou diminuée selon le niveau de la section.

exemple : si vous tapez :

```
.NP 1
Voici un exemple de paragraphe numéroté.Cette commande
est analogue à ".NH N" sauf qu'il n'y a pas d'intitulé et
que son numéro n'est pas souligné.
.NP 2
Voici un décalage
.NP 1
Valeur par défaut de N : 1
.NP 1
Un ".NP 0" remet la numérotation du premier niveau à 1.
```

cela donne :

1. Voici un exmple de paragraphe numèrotè.Cette commande est analogue à ".NH N" sauf qu'il n'y a pas d'intitulè et que le numèro n'est pas soulignè.

   1.1. Voici un dècalage

2. Valeur par dèfaut de N : 1.

3. Un ".NP 0" remet la numèrotation du premier niveau à 1.

3.2.3. Enumèration de paragraphes numèrotès ou precèdès d'un tiret.

- Si vous voulez ènumèrer une suite d'èlèments,tapez :

  " .TP " ("Tagged Paragraph")

  avant chaque èlèment de l'ènumèration.Ce dernier sera precèdè d'un tiret et dècalè automatiquement.

- Pour terminer votre ènumèration,tapez :

  " .RP " ("Return from Paragraph")

- Si on ne le fait pas,le formateur continue le texte à la suite du dernier èlèment.

- exemple : Cette sèrie de paragraphes a ètè obtenue en tapant
  .TP / texte / .TP / texte / .TP / texte / .TP / texte / .RP ("/" est mis pour le passage à la ligne)

1. De manière analogue,on peut obtenir une suite d'èlèments numèrotès.Chacun de ces paragraphes sera precèdè,non plus d'un tiret mais d'un numèro.

2. Avant chaque èlèment,tapez :

   " .NO " ("Numbered Notes")

3. Pour terminer cette ènumèration,tapez :

   " .NE " ("Numbered Notes End ")

4. exemple : C'est la suite d'opèrations qui ont ètè effectuèes pour obtenir ces quatre paragraphes.

3.3. Modification des marges,des alinéas.

3.3.1. .RL -Reset Level

Pour remettre à jour le niveau courant de l'alinéa,tapez :

" .RL "

3.3.2. .RS (Right Shift Start) - .RE (Right Shift End)

Pour augmenter la marge gauche de 5 positions,tapez :

" .RS "

Pour réduire la marge gauche de 5 positions,tapez :

" .RE "

3.3.3. exemple

Si vous tapez

.RS
Dans cet exemple,on a tapé .RS et voici la marge de gauche décalée vers la droite.Pour retourner à la position normale, on tape .RE à la fin de ce texte.
.RE
Et nous voisi revenus à la position normale de la marge de gauche;cette position tient compte,bien entendu,du niveau de section dans laquelle on se trouve.

vous obtenez :

Dans cet exemple,on a tapé .RS et voici la marge de gauche décalée vers la droite.Pour retourner à la position normale, on tape .RE à la fin de ce texte.
Et nous voisi revenus à la position normale de la marge de gauche;cette position tient compte,bien entendu,du niveau de section dans laquelle on se trouve.

3.4. Traitements particuliers de parties de texte.

3.4.1. .SQ (Squeeze Start) - .SE (squeeze End)

Il est parfois interessant de rètrècir un morceau de texte pour en accentuer la portèe.Cela se fait ,comme dans cet exemple-ci, à l'aide des commandes .SQ et .SE.

Pour commencer la contraction, tapez :

" .SQ "

Les marges droites et gauches des lignes suivantes seront augmentèes de 8 positions.Ce dècalage tient compte du dècalage courant et de la longueur de la ligne.

pour terminer la contraction,tapez :

" .SE "

Les dècalages et la longueur de la ligne sont remis à leurs anciennes valeurs.

3.4.2. .LS (Literal Insert Start) - .LE (Literal Insert End)

Si vous voulez prendre le texte tel qu'il est,c'est-à-dire sans le formater,tapez :

" .LS "

et terminez par :

" .LE "

3.4.3. .DS char (Display Start) - .DE (Display End)

Ces commandes entourent un texte que l'on dèsire faire passer dans le rèsultat sans transformation (si ce n'est un èventuel dècalage vers la droite 0.Elles permettent de mettre en èvidence un titre ou certaines phrases.

Si tel est votre dèsir,tapez :

".DS char"

où "char" est un des caractères suivants :

C - Centrer les lignes
L - justifier à gauche les lignes
I - dècaler les lignes
R - dècaler les lignes du double de la

valeur habituelle

Pour clôturer la portée de cette instruction,tapez,après avoir introduit votre texte :

" .DE "

exemple : Si vous tapez

```
.DS C
Et voici un exemple de
texte
centrè
automatiquement.
.DE
```

vous obtenez :

Et voici un exemple de
texte
centrè
automatiquement.

3.4.4. .SF (Start Figure) - .EF (End Figure)

Si vous voulez qu'une partie donnée de votre texte se trouve sur la même page,placez celle-ci entre les commandes

" .SF " et " .EF "

Ce texte n'est pas formatè.Il sera placè sur la page courante s'il y a assez de place.Sinon il apparaîtra au sommet de la page suivante.

3.4.5. Insertion de dessin

Si vous voulez conserver un espace libre pour y ajouter ultèrieurement un dessin à la main,tapez :

" .PI N "

où N est le nombre de lignes à passer sur la page. S'il n'y a pas assez de place sur la page courante,le texte continue sur cette page et l'insertion demandée se fera au dèbut de la page suivante.

3.5. .S N - Space N

Si vous voulez laisser un espace entre deux parties de texte,tapez :

" .S N "

où N est le nombre de lignes que vous désirez passer.
La ligne courante est automatiquement interrompue.

Cas particuliers :

- .S0 équivaut à .S 0 : on va à la ligne
- .S1 équivaut à .S 1 : on passe une ligne

3.6. .UL N - Underline

Pour souligner une phrase ou une partie de phrase,tapez :

" .UL N "

où N est le nombre de lignes à souligner.
Si vous ne faites suivre ".UL" d'aucun argument,la ligne suivante,uniquement sera soulignée (valeur par défaut de N = 1).

Cas particulier : pour ne souligner que quelques mots d'une phrase,vous écrivez sur la ligne qui suit ".UL",uniquement ces mots-là,et vous ajouterez la fin de la phrase à la ligne suivante.

3.7. .BP - Begin Page

Si vous voulez passer à la page suivante sans avoir complété celle qui est entamée,tapez :

" .BP "

Cette commande est utilisée,en particulier,pour les entêtes et les notes en bas de page.

4. STRUCTURE D"UN LIVRE OU D"UN ARTICLE

Dans cette section,nous considèrons un plus haut niveau de formatage. Il décide de toute la structure de l'article.Deux orientations de base sont possibles : le style "chapitre" utilisé pour les livres et les notes de cours et le style "article technique" . De plus,nous décrirons l'utilisation de notes infra-paginales.

4.1. Les trois parties du titre

Quelle que soit l'option choisie,il faut pour la mise en page des têtes de chapitre ou des intitulés d'articles,définir d'abord leur contenu.Une commande de titre place automatiquement trois champs respectivement à gauche, au centre et à droite de la ligne de titre.

Pour spécifier le contenu d'un titre,tapez :

" .TL sring-lh string-ch string-rh " (Title line)

où

- la chaîne de caractères représentée par "string-lh" sera justifiée à gauche dans une ligne de titre,celle représentée par "string-ch" sera centrée,tandis que celle représentée par "string-rh" le sera à droite.
- Chacune de ces chaînes doit être placée entre guillemets.
- N'importe quel de ces champs peut être vide (Si un d'entre eux est omis ,il faut néanmoins mettre les deux guillemets "" pour assurer la position exacte des différentes parties du titre (ceci n'est pas nécessaire pour le string-rh)).

4.2. Structure générale d'un livre

4.2.1. tête de chapitre - " .CP N string " - Chapter Titles

Pour définir des têtes de chapitres,tapez :

" .CP N string "

où "N" est le numéro du chapitre et "string" est son titre. Cette commande est précédée par une instruction ".TL ...".Son effet est

d'imprimer sur le côté gauche de chaque page "string-lh" (voir TL) et sur le côté droit le numéro du chapitre "N" concaténé avec le numéro de la page.

d'imprimer un peu plus bas sur la premiere page du chapitre,un titre centré de la forme

Chapitre N : string

Rien n'est imprimé en bas de la page.

L'option "Chapitre" peut être modifiée par .TT (voir plus loin).

4.2.2. Fin de chapitre - " .CE " - Chapter End

Cette commande est placée en fin de chapitre quand,dans le même fichier,on veut,immédiatement après commencer un autre chapitre.

4.2.3. Modification de l'option de base "chapitre"...

Si vous voulez modifier l'option "chapitre" pour une autre,telle que "annexe","chapter","appendix",.....,tapez

".ds TT string" (top Title)

où "string" est la nouvelle dénomination.
Cette commande doit être la première du texte.

4.2.4. exemple

Si vous tapez au début de votre chapitre :

```
.TL "ANALYSE" "" ""
.CP 2 "ANALYSE NUMERIQUE (1ère PARTIE)
```

voici la page 4 du chapitre 2 dont le nom abrégé est "ANALYSE" :

```
ANALYSE                                                    2.4
```

et voici la première page :

```
ANALYSE                                                    2.1

            Chapitre 2: ANALYSE NUMERIQUE (1ère PARTIE)
```

4.3. Structure générale d'un article

Pour définir un intitulé d'article technique,tapez ;

" .TR " (Technical Report Titles)

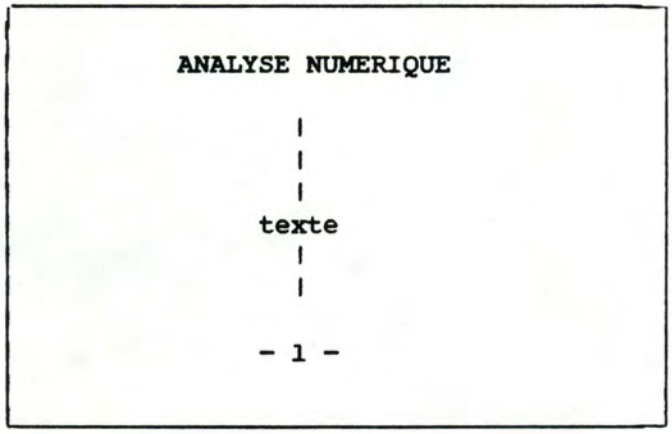
Cette commande doit être précédée de l'instruction ".TL...". Son effet est :

- d'imprimer au sommet de chaque page les trois champs définis dans l'instruction .TL
- d'imprimer au bas de la page,le numéro de la page (%) sous la forme -%-.

exemple : si vous tapez

```
.TL "" "ANALYSE NUMERIQUE" ""
.TR
```

vous obtenez comme première page



4.4. Notes infra-paginales

Si vous voulez placer des notes en bas en page,tapez :

" .FS " (Footnote Start)

Les lignes suivantes sont placées au bas de la page courante séparées du texte par une ligne.
Vous pouvez mettre plus d'une note en bas de page.

Quand le texte que vous vouliez y placer a été introduit,tapez :

" .FE " (Footnote End)

afin de continuer le traitement normal sur la page courante .

exemple :voici les instructions qui ont permis de placer le commentaire au bas de cette page :

```
.FS
Les notes infra-paginales permettent de clarifier,de
commenter,de référencier certaines notions d'un texte.
.FE
```

Les notes infra-paginales permettent de clarifier,de commenter,de référencier certaines notions d'un texte.

5. LISTE ALPHABETIQUE DES COMMANDES DE NROFF -MC

.BP		Begin Page
.CE		Chapter End
.CH	N	Centered Numbered Heading
.CP	N string	Chapter Titles
.DE		Display End
.DS	char	Display Start
.ds TT	string	Top Title
.EF		End Figure
.FE		Footnote End
.FS		Footnote Start
.LE		Literal Inset End
.LS		Literal insert Start
.NE		Numbered Notes Heading
.NH	N	Numbered Section Heading
.NO		Numbered Notes
.NP	N	Numbered Paragraph
.P1		New Paragraph (1)
.PI	N	Picture Insert
.PN	N	Page Number
.PP	N	New Paragraph
.RE		Right Shift End
.RL		Reset Level
.RP		Return from Paragraph
.RS		Right Shift Start
.S	N	Space N
.S0		Space Zero
.S1		Space One
.SE		Squeezed End
.SF		Start Figure
.SH		Unnumbered Section Heading
.SQ		Squeeze (Start)
.TL	string-lh string-ch string-rh	
		Title Line
.TP		Tagged Paragraph
.TR		Technical Report Titles
.UL	N	Underline

PARTIE 2 : LES MACROS-INSTRUCTIONS -MS DE NROFF

1. INTRODUCTION

Cette partie décrit brièvement quelques macros-instructions définies pour les utilisateurs dans /usr/lib/tmac.s.Elle possède en matière de présentation générale d'un document scientifique des possibilités que n'offre pas l'option"-mc" mais dispose néanmoins de choix moindres dans la manipulation même du texte.

2. LES COMMANDES DE BASE DE FORMATAGE

Voici tout d'abord les différentes possibilités offertes.Une description plus complete sera donnée à la section suivante.

2.1. création de titres (pour les sections)

Tous ces titres sont soulignés.
- intitulé non numéroté : .SH
- intitulé numéroté selon son niveau de section : .NH N

2.2. création de paragraphes

- création d'un nouveau paragraphe avec alinéa à la première ligne : .PP N
  [cas particulier : .P1]
- paragraphe sans alinéas : .LP N
- paragraphe décalé en bloc : .IP string N

2.3. Modifications des marges,des alinéas.

décalage de la marge de gauche

.RS : augmentation
.RE : diminution

2.4. traitements particuliers de parties de texte

- mise en évidence de lignes

  début de traitement : .DS
  fin de traitement : .DE

- insertion de texte continu

  début d'insertion : .KS ou .KF

fin d'insertion : .KE

2.5. Comment passer des lignes

.sp N [cas particuliers : .S0 et .S1]

2.6. souligner des lignes

.UL N

2.7. Pour passer a la page suivante

.br

2.8. double colonne

passage à deux colonnes : .2C
retour à une colonne : .1C

2.9. Date

date particulière : .DA
pas de date : .ND

2.10. Présentation générale : livre ou article technique

- définition du titre : .TL
- auteur : .AU
- institution de l'auteur : .AI
- resume :
  .AB : début du résumé
  .AE : fin du résumé
- notes infra-paginales

  début de la note : .FS
  fin de la note : .FE

3. DESCRIPTION DES COMMANDES DE FORMATAGE

Sont repris dans cette section uniquement les commandes non mentionnées dans la partie 1.

3.1. Création de paragraphes

3.1.1. .LP N - New Paragraph

Pour créer un nouveau paragraphe,tapez :

" .LP N "

où N est le nombre de lignes à passer avant le début de celui-ci.

exemple : Si vous tapez :

.LP 2
Vous observez qu'on passe 2 lignes avant de commencer ce nouveau paragraphe.

vous obtenez :

Vous observez qu'on passe 2 lignes avant de commencer ce nouveau paragraphe.

3.1.2. .IP string N - Indented Paragraph

Pour décaler un paragraphe en bloc,tapez

" .IP string N "

où N est la longueur de l'indentation
string la chaine de caractères à mettre devant le paragraphe décalé

exemple : si vous tapez :

.IP [1]
Texte du premier paragraphe,tapé normalement sur le nombre de lignes voulues.
.IP [2]
texte pour le second paragraphe

cela donne :

[1] texte du premier paragraphe,tapé normalement sur le nombre de lignes voulues.
[2] texte pour le second paragraphe

La séquence

```
.IP 1er: 9
L'étiquette plus large exige qu'on décale
plus le paragraphe
.IP deuxième:
et ainsi de suite
```

produit

```
1er:        L'étiquette plus large exige qu'on décale
            plus le paragraphe
deuxième:et ainsi de suite
```

3.2. Insertion d'un texte en un bloc

Si vous voulez qu'une partie donnée de votre texte se trouve sur la même page,placez celle-ci entre les commandes

" .KS " et " .KE "

Ce texte n'est pas format é.Il sera placé sur la page courante s'il y a assez de place.Sinon il apparaîtra au sommet de la page suivante.

On peut remplacer ".KS" par ".KF".Le résultat est identique sauf que le texte qui suit cette partie de texte sera copiée sur la page courante.Cette commande évite l'introduction de trop grands espaces blancs dans le document.

3.3. Double colonne.

Si vous placez la commande

.2C

dans votre document,celui-ci sera formaté en deux colonnes à partir de ce moment.Pour retourner à la présentation à une colonne,tapez

.1C

3.4. Date.

Les documents formatés avec l'option "-ms" possède une date au bas de chaque page.Pour forcer l'apparition d'une date précise en bas de page,tapez

.DA date

où date est la date à imprimer.
Pour supprimer l'apparition de la date en bas de page,tapez

.ND date

où date sera imprimée uniquement sur la page d'entête.

3.5. Présentation générale de l'article.

La page de couverture d'un document peut avoir une présentation particulière.La commande ".RP" le permet.La présence de cette dernière permet de définir le titre,l'auteur,l'institution,le résumé. La séquence d'instruction est la suivante :

```
.RP
.TL
titre du document (une ou plusieurs lignes).
.AU
auteur(s) (une ou plusieurs lignes)
.AI
institution(s) de l'auteur
.AB
abstract;résumé à placer sur la page d'entête de l'article
.AE (abstract end)
texte ...
```

Ces commandes sont facultatives.Néanmoins,pour supprimer le mot "abstract" ,il faut écrire ".AB no".

PARTIE 3: DIFFERENTES ECRITURES

Voici les quelques instructions qui permettent de changer de police de caractères et de passer aux caractères gras (quelle que soit la police utilisée).Cette liste peut s'agrandir avec le nombre de jeux de caractères disponibles sur la Sanders.

1. Caractères gras

Pour écrire une phrase ou une partie de phrase en **caractères gras** ,tapez :

".BO N"

où N est le nombres de lignes à prendre en considération.Si vous ne faites suivre ".BO" d'aucun argument,N aura la valeur 1 par défaut.

2. Draft

Pour écrire une phrase ou une partie de phrase en DRAFT (MEDIA MAX) ,tapez :

".D N"

où N est le nombres de lignes à prendre en considération.Si vous ne faites suivre ".D" d'aucun argument,N aura la valeur 1 par défaut.

3. Italique

Pour écrire une phrase ou une partie de phrase en *italique* ,tapez :

".I N"

où N est le nombres de lignes à prendre en considération.Si vous ne faites suivre ".I" d'aucun argument,N aura la valeur 1 par défaut.

4. Helvesan Regular

Pour écrire une phrase ou une partie de phrase en helvesan regular ,tapez :

".HR N"

où N est le nombres de lignes à prendre en considération.Si vous ne faites suivre ".HR" d'aucun argument,N aura la valeur 1 par défaut.

PARTIE 4 : EQUATIONS MATHEMATIQUES - MODE D'EMPLOI

$$G(z) = e^{\ln G(z)} = \exp\left[\sum_{k\geq 1} \frac{S_k z^k}{k}\right] = \prod_{k\geq 1} e^{S_k z^k/k}$$

$$= \left[1+S_1 z+ \frac{S_1^2 z^2}{2!}+\ldots\right]\left[1+ \frac{S_2 z^2}{2} + \frac{S_2^2 z^4}{2^2\ 2!} +\ldots\right]\ldots$$

$$= \sum_{m\geq 0}\left[\sum_{\substack{k_1,k_2,\ldots,k_m\geq 0 \\ k_1+2k_2+\ldots+mk_m=m}} \frac{S_1^{k_1}}{1^{k_1}k_1!}\,\frac{S_2^{k_2}}{2^{k_2}k_2!}\ldots\frac{S_m^{k_m}}{m^{k_m}k_m!}\right] z^m$$

1. INTRODUCTION.

NEQN est un programme qui permet d'introduire des textes mathématiques sur UNIX.Le langage NEQN a comme objectif d'être facile à utiliser pour les profanes en mathématiques.Par conséquent,il ne suppose connues,que relativement peu de notions . En particulier,les symboles mathématiques tels que +,-,*,parenthèses,... n'ont aucune signification particulière.

NEQN travaille comme un prèprocesseur pour le formateur de texte NROFF. En effet,la manière habituelle d'opérer est de préparer un document qui contient un mélange d'équations mathématiques et de texte normal. NEQN se charge des mathématiques tandis que NROFF se charge du corps du texte.

Pour formater des fichiers contenant des équations,tapez :

```
neqn nom(s) de fichier | nroff
```

2. LES EQUATIONS.

Les équations sont généralement imbriquées dans un texte.Pour spécifier à NEQN quand une expression mathématique commence et se termine,il faut l'entourer de lignes commençant par .EQ et .EN. Donc si vous tapez :

```
.EQ
x=y+z
.EN
```

vous obtiendrez en fin de compte :

x=y+z

Ce qui se trouve entre".EQ" et ".EN" est copié d'un bout à l'autre sans aucune modification ;Il n'y a que NEQN qui traite cette partie; pour centrer,numéroter ...,vous devez vous en préoccuper vous-même.Pour centrer une équation,par exemple,utilisez :

```
.ce
.EQ
a = b + c + d
.EN
```

vous obtiendrez :

a=b+c+d

Les macros "-MS" de nroff permettent,cependant, de centrer,de décaler,de justifier à gauche et de numéroter les équations.Avec cette option ,les équations sont centrées par défaut.Pour justifier à gauche une équation,utilisez .EQ L au lieu de .EQ.Pour les décaler un peu à droite,.EQ I.Chacune de ces expressions peut être suivie d'un numéro d'équation qui sera placé à la marge de droite.Par exemple,si on

introduit

```
.EQ I (3.1a)
x=f(y/2) + y/2
.EN
```

on obtiendra

$$x=f(y/2) + y/2 \qquad (3.1a)$$

Une notation abrégée permet d'inserer des expressions du style π_i^2 à l'intérieur d'une ligne ,sans utiliser .EQ et .EN. Elle sera décrite ultérieurement.

3. LES ESPACES BLANCS INTRODUITS.

NEQN diminue les espaces blancs et le passage à la ligne à l'intérieur d'une expression.Donc entre .EQ et .EN,les équations suivantes

```
x=y+z
et x   =       y +  z
et x  = y
        + z
```

...produisent toutes le même résultat :

```
x=y+z
```

Vous pouvez donc espacer et aller à la ligne à votre grè pour introduire vos équations de manière lisible et facile à éditer.Les longues lignes,en particulier,sont une mauvaise idée parce qu'elles ne sont pas toujours faciles à rectifier en cas d'erreur.

4. LES ESPACES BLANCS SORTIS.

Pour obtenir des blancs supplémentaires à la sortie ,utilisez un tilde par caractère blanc voulu.

exemple : si vous tapez

```
x~=~y~+~ z
```

vous obtiendrez

```
x = y + z
```

5. SYMBOLES,NOMS PARTICULIERS,GREC.

NEQN connaît quelques symboles mathématiques,quelques noms mathématiques et l'alphabet grec [1].Par exemple,

```
x = 2 pi int sin ( omega t)dt
```

produit

$$x=2\pi\int\sin(\omega t)dt$$

Ici les espaces blancs introduits à l'entrée sont nécessaires pour que NEQN sache que int,pi et omega sont des entités séparées qui nécessitent un traitement particulier.

Si vous hésitez,laissez des blancs entre les différentes parties des entrées.Une erreur courante est de taper f(pi) sans laisser de blancs de part et d'autre de "pi".Par conséquent,NEQN ne reconnaît pas "pi" comme étant un mot particulier, et en sortie,apparaît f(pi) à la place de $f(\pi)$.

6. INDICES ET EXPOSANTS.

Les indices et les exposants sont obtenus grâce aux mots sub et sup. En particulier,

```
x sup 2 + y sub k
```

donne

$$x^{2}+y_{k}$$

Les mots sub et sup doivent être entourés par des blancs;De plus,il ne faut pas oublier de laisser un espace (ou un tilde,etc) pour délimiter la fin d'un indice ou d'un exposant.L'erreur classique est d'écrire,par exemple,

```
y=(x sup 2)+1
```

ce qui donne

$$y=(x^{2})+1$$

au lieu de

$$y=(x^{2}) +1$$

---

[1] voir la liste complète à la fin de ce chapitre.

Les indices d'indices et les exposants d'exposants sont aussi admis:

```
x sub i sub 2
```

donne :

x_{i_2}

Un indice et un exposant qui porte sur le même élément sont imprimés l'un au-dessus de l'autre si l'indice est écrit en premier lieu :

```
x sub i sup 2
```

donne :

x_i^2

En dehors de ce cas particulier,si sub et sup sont groupés avec l'exposant d'abord,comme x sup y sub z ,cela donne x^{y_z} et non x_z^y.

7. ACCOLADES DE REGROUPEMENT.

Normalement,la fin d'un indice ou d'un exposant est délimitée simplement par un blanc (ou une tabulation ou un tilde ...) .Mais si dans l'indice ou l'exposant,il faut taper quelque chose qui contient des blancs? Dans ce cas,on utilise les accolades { et } pour marquer le début et la fin de l'indice ou de l'exposant :

```
e sup {i omega t}
```
donne $e^{i\omega t}$

REGLE : Les accolades peuvent toujours être utilisées pour obliger NEQN à traiter quelque chose comme une seule entité,ou simplement pour clarifier votre intention.

Ainsi,

```
x sub {i sub 1} sup 2
```
donne $x_{i_1}^2$

avec des accolades,mais

```
x sub i sub 1 sup 2
```
donne $x_{i_1}^2$

ce qui est plutôt différent.

On peut mettre , si nécessaire , des accolades à l'intérieur d'accolades :

```
e sup {i pi sup {rho + 1}}
```
donne $e^{i\pi^{\rho+1}}$

En règle générale,partout où vous pouvez utiliser des mots simples comme p.ex. x , vous pouvez utiliser une expression arbitrairement

compliquée si vous la mettez entre accolades.

Parfois vous devrez imprimer des accolades.Dans ce cas,il faut les mettre entre guillemets,comme "{" [1].

8. FRACTIONS.

Pour obtenir une fraction,utilisez le mot **over**

```
a+b over 2c = 1
```

donne

$$\frac{a+b}{2c}=1$$

La barre de fraction est de bonne longueur et positionnée automatiquement. Des accolades peuvent être utilisées pour clarifier "ce qui va" sur "quoi" :

```
{alpha + beta} over {sin(x)}
```

donne

$$\frac{\alpha+\beta}{\sin(x)}$$

NEQN considère **over** de priorité moindre [2] que sup et sub; par conséquent,

```
-b sup 2 over pi
```

donne

$$\frac{-b^2}{\pi} \qquad \text{et non pas} \qquad -b^{\frac{2}{\pi}}$$

Quand vous avez un doute, **utilisez les accolades** pour clarifier ce qui va avec quoi.

autres exemples

```
a+b over c+d+e = 1
```

donne

$$\frac{a+b}{c+d+e} = 1$$

---

[1] Cette notion de guillemets sera expliquée ultérieurement.

[2] Les lois de priorités des opérations sont résumées au paragraphe 22.

```
{partial sup 2 f} over {partial x sup 2} =
x sup 2 over a sup 2 + y sup 2 over b sup 2
```

donne

$$\frac{\partial^2 f}{\partial x^2} = \frac{x^2}{a^2} + \frac{y^2}{b^2}$$

9. RACINES CARREES.

Pour obtenir une racine carrée,utilisez le mot sqrt .

exemples

```
sqrt a+b    +    1 over sqrt {ax sup 2 +bx+c}
```

donne

$$\sqrt{a+b} + \frac{1}{\sqrt{ax^2+bx+c}}$$

```
sqrt {a sup 2 over b sub 2}
```

donne

$$\sqrt{\frac{a^2}{b_2}}$$

```
x = {-b +- sqrt{b sup 2 -4ac}} over 2a
```

donne

$$x=\frac{-b\pm\sqrt{b^2-4ac}}{2a}$$

```
x sup 2 over a sup 2    =    sqrt {pz sup 2 + qz + r}
```

donne

$$\frac{x^2}{a^2} = \sqrt{pz^2+qz+r}$$

```
x= {-b +- sqrt {b sup 2 -4 sqrt {ac}}} over sqrt {2a}
```

donne

$$x=\frac{-b\pm\sqrt{b^2-4\sqrt{ac}}}{\sqrt{2a}}$$

10. SOMMATION, INTEGRALE, ...

Les sommations, les intègrales et les constructions similaires sont aisées :

```
sum from i=0 to infinity x sub i = pi over 2
```

donne

$$\sum_{i=0}^{\infty} x_i=\frac{\pi}{2}$$

De même,

```
sum from i=0 to {i= inf} x sup i
```

donne

$$\sum_{i=0}^{i=\infty} x^i$$

remarque : l'utilisation des accolades indique le début et la fin de la partie supérieure "i= ∞". Pour la partie inférieure "i=0", ce n'est pas nécessaire parce qu'elle ne contient pas de blancs.
Si les parties from et to contiennent des blancs, vous devez les entourer par des accolades.

Les parties from et to sont toutes les deux optionnelles mais, si les deux sont utilisées, elles doivent l'être dans cet ordre.

D'autres caractères utiles peuvent remplacer le signe de sommation dans notre exemple :

```
int     prod    union   inter
```

deviennent, respectivement :

$$\int \qquad \prod \qquad \cup \qquad \cap$$

La construction from-to peut parfois être utilisée de manière inattendue :

```
lim from {n -> inf} x sup n =0
```

donne

$$\lim_{n\to\infty} x^n=0$$

Pour les bornes d'une intégrale,il est néanmoins plus élégant d'utiliser les notations envisagées pour les indices et les exposants. La construction aura donc la forme int sub ... sup ...

```
erf (z)    =    2 over sqrt pi int sub 0 sup z e sup -t sup 2 dt
```

donne

$$\mathrm{erf}(z) = \frac{2}{\sqrt{\pi}} \int_0^z e^{-t^2} dt$$

autres exemples

```
zeta (s)    =    sum from k=1 to inf
k sup -s        (Re s > 1)
```

donne

$$\zeta(s) = \sum_{k=1}^{\infty} k^{-s} \quad (\mathrm{Re}\, s > 1)$$

```
lim from {x -> pi / 2} (tan    x) sup {sin    x}    =    1
```

donne

$$\lim_{x \to \pi/2} (\tan x)^{\sin x} = 1$$

```
prod from i=0 to n a sub i =1
```

donne

$$\prod_{i=0}^{n} a_i = 1$$

11. DIACRITICAL MARKS.

Pour mettre des marques particulières sur le sommet des lettres,il existe quelques mots :

```
x dot      ẋ
x dotdot   ẍ
x hat      x̂
x tilde    x̃
x bar      x̄
x under    x̲
```

La marque est placée à la bonne hauteur.Les BAR et UNDER ont la longueur exacte de la construction entière; les autres marques sont centrées.

exemple : si vous tapez

```
x dot + X dot + y hat + y dotdot + x-y under + {alpha + beta} bar + x tilde
```

vous obtenez

$\dot{x}+\dot{X}+\hat{y}+\ddot{y}+\underline{x-y}+\overline{\alpha+\beta}+\tilde{x}$

12. TEXTE ENTRE GUILLEMETS.

Tout texte mis entre guillemets ("...") n'est soumis à aucun ajustement d'espace,comme NEQN a l'habitude de le faire.Ceci vous fournit une manière d'ajuster et d'espacer selon votre goût,si nécessaire :

```
" sin ( x ) " + sin(x)
```

fournit

sin (x) +sin(x)

Les guillemets sont aussi utilisés pour conserver des accolades ou pour imprimer tout autre mot qui a une signification pour NEQN :

```
"{alpha}"
```

donne

{alpha}

La construction "" est souvent utilisée pour conserver une place quand grammaticalement NEQN a besoin de quelque chose,mais que vous ne souhaitez rien à cet endroit dans votre texte de sortie.

Par exemple, pour obtenir ^{2}He ,vous ne pouvez pas n'écrire que "sup 2 He" parce qu'un sup doit être l'exposant de quelque chose.Par conséquent,vous devez dire

```
"" sup 2 He
```

Pour pouvoir imprimer un guillemet,il faut le précéder de \,c-a-d écrire \".

13. ALIGNEMENT DES EQUATIONS

IL est parfois nécessaire d'aligner un terme de diverses équations en position horizontale,entre autres,le plus souvent le signe d'égalité.Les deux opérations **mark** et **lineup** le permettent.

Le mot **mark** peut apparaître une fois à n'importe quel endroit dans une équation.Il enregistre la position horizontale où il est apparu.Des équations successives peuvent contenir une occurrence du mot lineup. L'endroit où **lineup** apparaît est aligné sur la place marquée par le **mark** précédent,si c'est possible.

exemple

Vous pouvez taper :

```
.EQ
x+y mark = z
.EN
.SO
.EQ
x lineup = 1
.EN
```

et vous obtiendrez :

$$x+y=z$$
$$x=1$$

Remarque

Vous ne pouvez pas écrire

```
x mark = 1
...
x + y lineup = z
```

parce qu'il n'y a pas de place pour mettre la partie $x+y$ vu l'endroit où est positionné le **mark**.

autre exemple

```
.EQ
f(x) mark = y
.EN
.SO
.EQ
lineup = z
.EN
```

donne

$$f(x)=y$$
$$=z$$

14. LES GRANDES PARENTHESES,...

Pour obtenir des grands crochets [ ],accolades { },parenthèses ( ),et des traits verticaux | autour de certaines expressions,utilisez les commandes : **left** et **right**

Par exemple,si vous tapez :

```
left { a over b+1 right} = left ( C over d right ) + left [ e right ]
```

vous obtenez

$$\left\{\frac{a}{b+1}\right\}=\left(\frac{C}{d}\right)+[\ e]$$

Les crochets obtenus sont suffisamment grands pour couvrir tout ce qu'ils entourent.

remarques

1. Les accolades sont plus grandes que les crochets et les parenthèses, parce qu'elles sont construites avec 3,5,7,... pièces tandis que les crochets peuvent l'être avec 2,3,....

2. La partie **right** est optionnelle : un **left**... ne nécessite pas un **right**...

3. Si vous désirez omettre la partie **left** ,c'est un peu plus compliqué parce que techniquement,vous ne pouvez avoir un **right** sans un **left** correspondant.Mais vous pouvez,par exemple utiliser l'astuce suivante :

```
left "" ... right )
```

Le **left** signifie que vous avez un left de rien du tout.Ceci répond à la grammaire et ne gêne pas votre sortie.

autre exemple

Si vous tapez

```
left [ x+y over 2a right ]     =     1
```

cela donne

$$\left[\frac{x+y}{2a}\right] = 1$$

15. LES PILES.

Il existe un procédé général pour empiler verticalement des objets.Par exemple,

```
A     =     left [
pile { a above b above c}       pile {x above y above z}
right ]
```

donne

$$A = \begin{bmatrix} a & x \\ b & y \\ c & z \end{bmatrix}$$

Les èlèments d'une **pile** (aussi nombreux que vous le voulez et que la mèmoire le permet !) sont centrès les uns au-dessus des autres.Le mot-clè **above** est utilisè pour sèparer les diffèrents morceaux.

Il existe trois autres formes de piles :

lpile : pile dont les èlèments sont justifiès à gauche
rpile : pile dont les èlèments sont justifiès à droite
cpile : pile centrèe (comme pile).

L'espacement vertical entre les èlèments est un peu plus grand dans les l-,r- et cpiles qu'il ne l'est pour les piles ordinaires.

autre exemple

```
sign(x)     ==     left {
   rpile {1 above 0 above -1}
      lpile {if above if above if}
      lpile {x>0 above x=0 above x<0}
```

donne

$$\text{sign}(x) \equiv \begin{cases} 1 & \text{if } x>0 \\ 0 & \text{if } x=0 \\ -1 & \text{if } x<0 \end{cases}$$

16. LES MATRICES.

Il est aussi possible d'écrire des matrices.Par exemple,pour obtenir un tableau net tel que

$$x_i \quad x^2$$

$$y_i \quad y^2$$

vous devez taper

```
matrix {
  ccol {x sub i above y sub i}
  ccol {x sup 2 above y sup 2}
```

Ceci produit une matrice à deux colonnes centrées.Les éléments des colonnes sont enregistrés de la même manière que pour une pile, chaque élément est séparé par le mot **above**. Vous pouvez aussi utiliser **lcol,rcol** pour ajuster à gauche ou à droite dans les colonnes.Chacune d'elles peut être ajustée séparément et il peut y avoir autant de colonnes que vous le désirez.

L'utilisation d'une matrice à la place de 2 piles adjacentes,permet,si les éléments des piles n'ont pas tous la même hauteur,de les aligner correctement.Une matrice les force à s'aligner parce qu'elle regarde d'abord la structure entière avant de décider de l'espacement utilisé.

ATTENTION :chaque colonne de la matrice doit avoir le même nombre d'éléments.

autre exemple : si vous tapez

```
left | left |
matrix { lcol { a above b} ccol {c above d} rcol {e above f}}
right | right |
```

cela donne

$$\left\|\begin{matrix} a & c & e \\ b & d & f \end{matrix}\right\|$$

17. INTRODUCTION ABREGEE DES EQUATIONS.

Dans un document mathématique,il faut suivre les conventions mathématiques non seulement dans les équations isolées mais aussi à l'intérieur même du texte,par exemple,en mettant le nom des variables grecques comme α. Ceci peut être effectué en entourant les parties appropriées avec .EQ et .EN.Seulement,la répétition continuelle de .EQ et .EN est fastidieuse. De plus,avec l'option "-MS",.EQ et .EN sousentendent une équation isolée.

NEQN permet d'introduire des courtes expressions dans une ligne grâce à une notation abrégée.Vous pouvez définir deux caractères pour marquer le début et la fin d'une équation dans une ligne et taper,dès lors,des expressions parmi des lignes de texte.

EXEMPLE.

definition de délimiteurs.

Si vous mettez les caractères gauche et droit au signe dollar,ajoutez au début de votre document :

```
.EQ
delim $$
.EN
```

utilisation des delimiteurs.

Ceci etant fait,vous pouvez alors ecrire des phrases du style :

```
Let $alpha sub i$ be the primary variable,and let $beta$ be
zero.Then we can show that $x sub 1$ is $>=0$ .
```

ce qui vous donnera :

Let α_i be the primary variable,and let β be zero.Then we can show that x_1 is ≥ 0 .

mise hors de service des delimiteurs.

```
.EQ
delim off
.EN
```

remarques :

Comme vous vous en doutez,les espaces,les passages à la ligne,... ont leur importance dans le texte,mais pas dans la partie equation elle-même. Il peut y avoir plusieurs equations sur une même ligne d'entree.

Il faut laisser suffisamment d'espace avant et après une ligne qui contient des expressions telles que $\sum_{i=1}^{n} x_i$ afin qu'il n'y ait pas d'interferences avec les lignes qui l'entourent.

Ne pas utiliser des accolades,des accents circonflexes ou des guillemets comme delimiteurs.

18. DEFINITIONS.

18.1. Utilisation frequente d'une même sequence de caracteres.

NEQN permet de donner un nom à une sequence de caracteres frequemment utilisee.Il suffira juste,ensuite,de taper ce nom au lieu de toute la chaine.

exemple : Si la sequence

```
x sub i sub 1 + y sub i sub 1
```

apparait plusieurs fois dans un texte,vous pouvez ne la tapez qu'une fois en la definissant de la manière suivante :

```
.EQ
define xy "x sub i sub 1 + y sub i sub 1"
.EN
```

et ensuite,à chaque occurrence de cette expression,vous la remplacerez par le nom que vous lui avez donnè.Par exemple,en tapant

```
.EQ
f(x)= { xy } over 2a
.EN
```

vous obtenez

$$f(x)=\frac{x_{i_1}+y_{i_1}}{2a}$$

Et ainsi,chaque occurrence xy sera remplacée par la valeur dèfinie antèrieurement.Il faut nèanmoins faire attention à laisser des blancs (tildes,...) autour de ce nom quand vous l'utilisez,pour que NEQN le reconnaisse comme ètant spècial.

remarque : il est interdit de dèfinir un èlement en terme de lui-même.

18.2. Redefinition de "mots-cle".

Vous pouvez redèfinir des mots significatif de NEQN;par exemple,si vous voulez que "/" ait la signification de "over",tapez :

```
define / "over"
```

Dès lors,si vous tapez

```
f(x)= a sup 2 / b sup 2
```

vous obtenez bien

$$f(x)=a^2/b^2$$

Pour redèfinir "over" comme ètant "/",vous devez ècrire :

```
define "over" "/"
```

afin èviter une erreur de syntaxe.

19. MOUVEMENTS LOCAUX.

Bien que NEQN tâche de placer les entitès le mieux possible sur la page,ce n'est pas nècèssairement comme vous l'auriez souhaitè.Des commandes sont prèvues et permettent des dèplacements latèraux et verticaux :

```
FWD  N : avancer de N positions
BACK N : reculer de N positions
UP   N : monter de N positions
```

DOWN N : descendre de N positions

20. UN EXEMPLE.

Voici la sèquence d'instructions qu'il a fallu introduire pour obtenir l'équation de la première page.

```
.EQ
G(z) mark = e sup { ln G(z) }
= exp left (
sum from {k >= 1} {{S sub k z sup k} over k} right )
= prod from {k >= 1} e sup {S sub k z sup k /k}
.EN
.EQ
lineup = left (+ S sub 1 z +
{S sub 1 sup 2 z sup 2 } over 2! + ... right )
left ( {S sub 2 z sup 2 } over 2
+ {S sub 2 sup 2 z sup 4 } over { 2 sup 2 2!}
+ ... right )..
.EN
```

```
.EQ
lineup = sum from {m >= 0} left (
sum from
pile {k sub 1 ,k sub 2 ,...,k sub m >=0
above
k sub 1 +2k sub 2 + ... +mk sub m =m}
{S sub 1 sup {k sub 1}} over {1 sup k sub 1 k sub 1 !}
{S sub 2 sup {k sub 2}} over {2 sup k sub 2 k sub 2 !}
{S sub m sup {k sub m}} over {m sup k sub m k sub m !}
right ) sup m
.EN
```

21. MOTS CLE,PRIORITES,...

21.1. Priorités des opérations

Si vous n'utilisez pas d'accolades,NEQN effectuera les opérations dans l'ordre donné par cette liste[1] :

vec	under	bar	tilde	hat	dot	dotdot
fwd	back	down	up			
sub	sup	sqrt	over			
left	right					
from	to					

Ces opérations se groupent vers la gauche :

---

[1] NEQN connaît ces mots aussi bien en majuscules qu'en minuscules.

over sqrt left right

Toutes les autres se groupent vers la droite.

21.2. Mots-clé

Une série de caractères particuliers peuvent être introduits [1]. Ces séquences de caractères sont reconnues et modifiées de la manière suivante :

>=	≥
<=	≤
==	≡
!=	≠
+-	±
inf	∞
partial	∂
approx	≅
cdot	·
times	x
del	∇
sum	∑
prod	∏
union	∪
inter	∩
inc	⊂
comp	⊃
app	∈
cont	∋
ninc	⊄
ncomp	⊅
napp	∉
ncont	∌
pourtout	∀
ilexist	∃
nilexist	∄
/	∧
<|	⟨
>|	⟩
dbar	∥
inter	∩
int	∫

21.3. Lettres grecques

Pour obtenir des lettres grecques, écrivez en toutes lettres en majuscules ou en minuscules selon ce que vous désirez :

DELTA	Δ	iota	i
GAMMA	Γ	kappa	k
LAMBDA	Λ	lambda	λ
OMEGA	Ω	mu	μ
PHI	Φ	nu	?
PI	Π	omega	ω
PSI	Ψ	omicron	o
SIGMA	Σ	phi	φ
THETA	Θ	pi	π
UPSILON	U	psi	ψ
XI	X	rho	ρ
alpha	α	sigma	σ
beta	β	tau	τ
chi	X	theta	θ
delta	δ	upsilon	u
epsilon	ϵ	xi	ξ
eta	η	zeta	ζ
gamma	γ		

21.4. Mots connus de NEQN.

Voici la liste de tous les mots connus [1] par NEQN (sauf pour les caractères que l'on appelle par leur nom) ainsi que la section dans laquelle on en a parlé :

above	15,16	lpile	15
back	19	mark	13
bar	11	matrix	16
ccol	16	over	8
col	16	pile	15
cpile	15	rcol	16
define	18	right	14
delim	19	rpile	15
dot	11	sqrt	9
dotdot	11	sub	6
down	19	sup	6
from	10	tilde	11
fwd	19	to	10
hat	11	under	11
lcol	16	up	19
left	14	~	4
lineup	13	{ }	7
		"..."	7,12

---

[1] NEQN connaît ces mots aussi bien en majuscules qu'en minuscules.

22. ERREURS.

Si vous faites une erreur dans une équation,telle qu'un oubli d'accolade, une accolade de trop ou un "sup" sans rien devant,NEQN vous affichera le message

syntax error between lines x and y,file z

où x et y sont les lignes aux environs desquelles l'erreur est commise et z est le nom du fichier en question.Si vous essayez d'executer NEQN sur un fichier inexistant,vous recevrez à nouveau un message.

Les équations introduites peuvent être assez grandes.Ceci est dû au fait que NROFF possède une mémoire interne.Si vous recevez un message

line overflow

,vous avez dépassé la longueur de cette mémoire.La seule possibilité est alors de couper l'équation en deux parties.

NEQN ne prête aucune attention au fait que l'équation soit trop longue pour l'imprimer sur une ligne.

PARTIE 5: LES TABLEAUX

TBL est un programme qui permet d'introduire des tableaux sur UNIX.Il travaille comme un prèprocesseur du formateur Nroff.

Pour introduire un tableau dans le texte,il faut l'entourer par les lignes de commandes ".TS" et ".TE".La premiére ligne aprés ".TS" spècifie les diffèrentes colonnes de la table.Ceci consiste en une liste de descripteurs sèparès par des espaces blancs ou des tabulations.Chaque descripteur de colonne est une suite de caractères composè des lettres "n","r","c"."l" et "s".Ces diffèrentes lettres signifient :

- c centrer à l'intèrieur de la colonne.
- r justifier à droite dans la colonne.
- l justifier à gauche dans la colonne.
- n ajuster numèriquement : les unitès de chaque nombre sont alignées.
- s poursuivre le contenu de la colonne prècèdente sur cette colonne.

Le descripteur de colonne peut être suivi par un entier donnant le nombre d'espaces entre cette colonne et la suivante;on a 3 par dèfaut.

exemple : le descripteur "ccr5" indique que
- les 2 premières lignes de cette colonne sont centrées;
- la troisième et les lignes restantes sont justifiées à droite;
- et la colonne est séparée de la colonne de droite de 5 espaces blancs.

Pour obtenir le tableau

Town	Household Population Households	
	Number	Size
Bedminster	789	3.26
Bernards Twp.	3087	3.74
Bernardsville	2018	3.3
Bound Brook	3425	3.104
Branchburg	1644	3.49
Bridgewater	7897	3.81
Far Hills	240	3.19

il a fallu introduire les instructions suivantes,où \t est mis pour une tabulation :

```
.TS
cccl sccn sscn
Household Population
Town\tHouseholds
\tNumber\tSize
Bedminster\t789\t3.26
Bernards Twp.\t3087\t3.74
Bernardsville\t2018\t3.3
Bound Brook\t3425\t3.104
```

```
Branchburg\t1644\t3.49
Bridgewater\t7897\t3.81
Far Hills\t240\t3.19
 .TE
```

ANNEXE 4: LES JEUX DE CARACTERES ET LA LISTE DES COMMANDES DE LA SANDERS.

Dans cette annexe sont repris les différents jeux de caractères disponibles sur la Sanders actuellement,c-a-d

#0	(@) Messenger 12 Germany	630169
#1	(A) Media Max 12 Germany	630168
#2	(B) Greek Mathematics	630204
#3	(C) Scientific Pi	630208
#4	(D) Helvesan Italic ASCII	630159
#5	(E) Messenger 12 French	
#6	(F) Media Max 12 French	
#7	(G) Helvesan Regular ASCII	630204

ainsi que la liste des commandes Sanders qui permettent à l'utilisateur d'agir sur l'imprimante.

Messenger 12 ASCII 630149

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ abcdefghijklmnopqrstuvwxyz
1 2 3 4 5 6 7 8 9 0 - = ` [ ] ; ' " , . /
! @ # $ % ^ & * ( ) _ + ~ { } : " | < > ?
```

Media Max 12 ASCII 630151

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ ABCDEFGHIJKLMNOPQRSTUVWXYZ
1 2 3 4 5 6 7 8 9 0 - = ` [ ] ; ' " , . /
! @ # $ % ^ & * ( ) _ + ~ { } : " | < > ?
```

Scientific Pi 630208

```
ᶠ꜀ᵎᶠ꜀ᵢ≪≮≲≁≅±≮≡≤π∠⋄≮UC∇→∂⊥∝  ᵡ꜀ᵢᵎ+»≯≳≠≈∓}≅≥π⊥≏ℓ∩⊃Δ←X‖∞
1 2 3 4 5 6 7 8 9 0 ⁻ -   □ ) + 7 2 × + ∤
1   3 4 5 ↑ 6 = 8 9 † +   0 ( = 2 ( × +
```

Greek Mathematics 630204

```
ΑΒΓΔΕ∑ΖΗΘΙΚΛΜΝΞΟΠΡΣΤΥΦ∏Χ΍Ω αβγδϵϖζηθικλμνξοπρστυφχψω
1 2 3 4 5 6 7 8 9 0 - = √ [ ] ° ' " . . /
ᴦ @ # $ % ∧ ∣ ᒧ ( ) ¯ + ~ { } : " ∣ < > £
```

Helvesan Regular ASCII 630154

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ abcdefghijklmnopqrstuvwxyz
1 2 3 4 5 6 7 8 9 0 - = ` [ ] ; ' " , . /
! @ # $ % ^ & * ( ) _ + ~ ´ ¨ : " . < > ?
```

Helvesan Italic ASCII 630159

```
.ABCDEFGHIJKLMNOPQRSTUVWXYZ abcdefghijklmnopqrstuvwxyz
1 2 3 4 5 6 7 8 9 0 - = ` [ ] ; ´ \ , . /
! @ # $ % ^ & * ( ) _ + ~ { } : " ! < > ?
```

CHAPTER 8
SOFTWARE INTERFACE

The MEDIA 12/7 responds to commands sent to it from the host computer. These commands direct the printer to perform various functions. It is the responsibility of the applications or systems programmer to integrate these commands into his software so that the desired features of the MEDIA 12/7 can be utilized. This chapter contains a detailed explanation of each command. Table 8-1 lists the commands, the code that corresponds with each command and its default value.

PRINTER COMMANDS

The Sanders Technology MEDIA 12/7 Typographic Printer offers the user a high degree of control over the various printing functions provided by the printer. By specifying a particular sequence of control characters, the user can change default parameter settings, operate the basic print mechanism, or any number of other functions. A command language has been created in order to implement these operations. Table 8-1 provides a summary of these commands.

TABLE 8-1. MEDIA 12/7 Commands

Command	Code	Default Value
Standard Printer Commands		
Space	<SP>	See Note 2
Backspace	<BS>	See Note 2
Carriage Return	<CR>	
Force Print	<ESC>P	
Line Feed	<LF>	
Horizontal Tab	<HT>	
Form Feed	<FF>	
Set/Clear Underline	<ESC>u(n)	
Repeat Character	<ESC>s(n)(a)	
Insert Sequence Of Characters	<ESC>x(n)(aa...a)	
Horizontal Spacing Commands		
Space	<SP>	See Note 2
Backspace	<BS>	See Note 2
Insert Space	<ESC>y	
Force Escapemen'	<ESC>l(nn)	
Non-Escape	<ESC>N	
Tab Commands		
Horizontal Tab	<HT>	
Tab Set List	<ESC>p(nn),(nn),...,(nn).	
Tab Clear List	<ESC>q(nn),(nn),...,(nn).	
Clear All Horizontal Tabs	<ESC>2	
Absolute Horizontal Tab to Column #	<ESC>h(nn)	

TABLE 8-1. MEDIA 12/7 Commands (cont)

Command	Code	Default Value
Vertical Spacing Commands		
Line Feed	<LF>	
Half Line Feed	<ESC>U	
Negative Line Feed	<ESC><LF>	
Negative Half Line Feed	<ESC>D	
Force Leading (Mils)	<ESC>o(nn)	
Force Leading (Inches)	<ESC>O(n)	
Absolute Vertical Tab to Line #	<ESC>v(nn)	
Text Format Commands		
Select Format	<ESC>n(a)	Ragged Right
Set Line Length	<ESC>e(nnn)	7.0 inches
Set Left Margin	<ESC>f(nnn)	1.0 inch
Set Top Margin	<ESC>T(nnn)	0.5 inch
Set Botton Margin	<ESC>B(nnn)	0.5 inch
Vertical Margin Enable/Disable	<ESC>V(n)	Enable
Set Hanging Indent	<ESC>g(nnn)	0.0 inch
Set Single Line Indent	<ESC>I(nnn)	0.0 inch
Set Line Height	<ESC>k(nn)	See Note 1
Set Column Width	<ESC>C(nnn)	See Note 3
Set Letter Space	<ESC>j(nn),(nn).	See Note 1
Set Word Space	<ESC>w(nn),(nn),(nn).	See Note 1
Set Form Length	<ESC>i(nnn)	11.0 inches
Font Commands		
Assign Logical Font	<ESC>d(n)(font number)	
Select Logical Font	<ESC>a(n)	
Miscellaneous Commands		
Rounded Line Feed	<ESC>K(n)	Enabled
Select Draft Mode	<ESC>t	
Initialize Printer	<ESC><SUB>I	
Bolding	<ESC>b(n)	
Set Ribbon Usage	<ESC>r(n)	2

NOTES: 1. Default values are stored with each font. When a font is selected, the stored values are copied into the appropriate memory locations. All values are in units, or multiples, of mils. If the value for a command that has a default value is altered, the default value is reinstated when the printer in initialized (<ESC><SUB>I) or the printer is manually reset (RESET pushbutton).

2. Some of the standard printer commands are repeated in the table for purposes of clarity.

3. Upon selecting a font, the width of the space character for that font is instated as the column width. A Set Column Width command can be used to override this value.

Printer Command Language

The MEDIA 12/7 printer command language consists of two types of control character sequences. The first utilizes the standard control character set found in most ASCII printers. These are single character commands. The second type of control sequence is implemented through the use of an escape sequence. Each escape sequence is formed by using the ASCII ESC character (signified by <ESC> in this manual) followed by a code specifying the type of function to be performed and, if required, the specific values for the command arguments. Figure 8-1 is a definition of the escape sequence.

<ESC>FunctionArgument1Argument2 . . . Argumentn

Figure 8-1. Escape Sequence Definition

The argument(s) contained in the escape sequences is specified either as an alphabetic character or as a numeric value. Alphabetic characters are specified directly and are represented in this manual by (a).

The numeric values that are specified in the escape sequences are in the form of binary data. Binary data is represented by an (n) where "n" is a six-bit binary value. In order to pass these six-bit binary values to the printer, the binary data is made to look like normal 7-bit ASCII character data. The binary data is grouped in six-bit groups. This allows the highest order bit of the seven data bits to be turned "on" to "1". In this way any bit pattern in the following six bits will combine with the "on" bit to yield a printable ASCII character. Whenever the notation (n) is used in the following text, it is understood that there are six bits and one "on" bit. Negative numeric values are specified as the two's complement of the binary value.

In some commands, it is necessary to specify a numeric value larger than one that can be represented in a single six-bit segment (63 decimal). In these cases, two or three six-bit segments must be specified. This is necessary in order to use the printer with seven-bit non-transparent ASCII data. As stated above, (n) allows the representation of a binary value within the range of -64 to +63. Likewise, (nn) allows the range -2048 to +2047. The largest value that can be represented is (nnn) where the range -32768 to +32767 is available. Due to printer restrictions, only the first sixteen bits of the available eighteen are used.

For example, the escape sequence to specify a line length of 6 inches (6000 mils) is:

<ESC>e(n1n2n3)

where:	n1 =	0	1	0	0	2^{15}	2^{14}	2^{13}	2^{12}
	n2 =	0	1	2^{11}	2^{10}	2^{9}	2^{8}	2^{7}	2^{6}
	n3 =	0	1	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}

To specify 6000 mils, it is necessary to use the following powers of 2:

2^{12} = 4096

2^{10} = 1024

2^{9} = 512

2^{8} = 256

2^{6} = 64

2^{5} = 32

2^{4} = 16

6000

Therefore: n1 = 01000001 = 41 (hex) = A (ASCII)

n2 = 01011101 = 5D (hex) =] (ASCII)

n3 = 01110000 = 70 (hex) = p (ASCII)

and the line length is specified as:

<ESC>eA]p

Standard Printer Commands

SPACE - <SP>

The space command causes the printhead to advance across the current line. The amount of movement is determined by the Space character of the font in use. Consecutive Space characters are not compressed when text is justified. Each space specified is retained through all formatting operations.

If the end of a sentence coincides with the end of a word-wrapped line, both spaces (if two are used to separate sentences) will be deleted. Therefore, no extraneous space characters will be included in the justified text.

BACKSPACE - <BS>

The Backspace command causes the printer to back the width of the space character of the current font. (Cannot be used to center one character over another when using a proportional-spacing font - see Non-Escape.) Only one backspace command can be specified between any two printable characters.

CARRIAGE RETURN - <CR>

The Carriage Return command forces the printer to terminate the line of text at the point the <CR> is entered.

The carriage return does not function like a carriage return in a conventional typewriter. Being a true bi-directional printer, printing can start from either end of a print line (exception: when the printer seeks and locates the left edge of the paper, printing will start from the left end of the print line). When a Carriage Return command is issued, the printer determines which end of the print line is closer to its current position and starts printing the next line from that end. This results in minimum printhead movement and, hence, a higher print speed.

Dip Switch 2 (either on the ROM/RAM board or on the rear interface panel) ca be set to instruct the printer to perform an automatic line feed in conjunctio with the Carriage Return or Force Print commands. If Dip Switch 2 is OPE (OFF), no line feed is performed. The CLOSED (ON) position instructs the printe to perform an automatic line feed.

FORCE PRINT - <ESC>P

The Force Print command causes the line of text preceding the command to be printed in the current format. This command is useful when it is desired to justify a proportional spacing font and to keep track of the number of lines created. Adding the character widths for each character used in the line of text and inserting a Force Print command after the last word (or word segment when the text is hyphenated) that will fit within the line length,causes the line to be printed in the justified format chosen. The number of lines created can then be counted. The Force Print command functions the same as the Carriage Return except the Carriage Return causes a line to be printed out in ragged right format without interletter of interword spacing. (Note: Character width specifications for each proportional spacing font are available from Sanders Technology upon request.)

LINE FEED - <LF>

The Line Feed command advances the paper to the next print line. The amount of advance movement (leading) is established by the Line Height that has been set for the current font. Each time the printer receives a Line Feed command, it will advance the paper one Line Height. The default Line Height can be altered using the Set Line Height (<ESC>k(nn)) command. The Line Feed can be included at any point in the text where a line feed is desired. It does not have to be preceded by a Carriage Return <CR> command.

After a line is printed, the paper is advanced to the next print line. The amount of advancement is the largest line height of the fonts used in the line just printed. If the Line Feed command is entered alone, no horizontal displacement of text occurs.

HORIZONTAL TAB - <HT>

This command moves the printhead to the next tab stop to the right. Tabs are set and cleared using the Tab Set List (<ESC>p(nn,nn,...,nn)), Tab Clear List (<ESC>q(nn,nn,...,nn), and Clear All Horizontal Tabs (<ESC>2) commands. (See also the Absolute Horizontal Tab to Column (<ESC>h(nn)) command.)

FORM FEED - <FF>

The Form Feed command is used to move the paper so that the printing mechanism is at the top of the next form. Upon receiving a Form Feed command, the printer advances the paper to the end of the sheet. The printer contains a cumulative leading counter that keeps track of the vertical position of the paper (as determined by the default or specified Form Length). When the Form Feed command is issued, the paper is advanced the remaining distance of the form and the cumulative leading counter is reset. The printer is then set to start printing (spacing to the specified head-of-form) the next sheet. The Form Feed command works the same for the standard manual sheet feeder and the optional sheet feeders. To alter the default Form Length (11 inches), use the Set Form Length (<ESC>i(nnn)) command. The Form Feed command can be included at any point in the text where a from feed is desired. It does not have to be preceded by a Carriage Return <CR> command.

SET/CLEAR UNDERLINE - <ESC>u(n)

The Set/Clear Underline command is used to initiate and terminate underlining. Underlining starts at the first <ESC>u command and terminates at the second <ESC>u command.

The value (n), expressed as an ASCII value representing a numeric value in the range 0 to 63, is the vertical displacement (in 3.5 mil increments) of the underline character. It must be included in the command. For example, if (n) is 11 (ASCII equivalent K), the underline character is displaced 38.5 mils below its normal position. The leading between print lines is not altered when the underline command is specified: care must be taken that the underline is not displaced into the next print line. Leading is not changed so that a constant white space is maintained between print lines. If different displacement values are specified in the first and second <ESC>u commands, the value specified in the second will be used. If several <ESC>u commands are processed at the same time, the value specified in the last command controls the vertical displacement. (Hint: Use the same displacement for all underlining.)

REPEAT CHARACTER - <ESC>s(n)(c)

The Repeat Character command causes the character (c), either an alphabetic or numeric character, to be printed (n) times. This command is useful to repeat a single character more than four or five times. When a character is repeated using this command, some data compression may be achieved

INSERT SEQUENCE OF CHARACTERS - <ESC>x(n)(cc...c)

This command causes the specified data (cc...c), which is (n) characters in length, to be repeated as many times as is necessary to fill the remainder of the print line. The Insert Sequence of Characters can be used only with ragged right text (Text Format 0) and the print line must be terminated by a Carriage Return (CR) command. The print line length is determined by the value of Line Length (default of specified). The print line consists of 1) preceding text, 2) sequence of characters and 3) subsequent characters (if any). Only one <ESC>x command can be contained in a line of text (i.e., between Carriage Return (CR) commands.

For example:

CHAPTER 1<ESC>xB .1-1<CR>

will print out as:

CHAPTER 1 1-1

The B in the Insert Sequence of Characters command indicates a sequence length of two characters. The characters are space and period.

To print the same line but with a space before the page number, specify the following:

CHAPTER 1<ESC>xB . 1-1<CR>

The B indicates a two character sequence (space and period) followed by the four characters (space one dash one). This will print out as:

CHAPTER 1 1-1

If one sequence of characters cannot fit in the space available, characters, starting with the leftmost character, are eliminated until the sequence fits.

If one or more sequence of characters can fit, a whole number of the specified character sequences is inserted starting from the right and filling in to the left. No character elimination is performed. If an entire sequence cannot fit, an appropriate number of blank spaces are inserted in its place. For example, the space available contains room for 17 characters and the character sequence contains three characters. The insert will contain five complete sets with two blank spaces to the left of the printed sequences.

Horizontal Spacing Commands

The Space and Backspace commands are detailed in "Standard Printer Commands" above. The other horizontal spacing commands are included in this section. No more than 20 horizontal spacing commands can be included in one print line.

INSERT SPACE - <ESC>y

This command causes sufficient white space to be inserted between two blocks of text to fill the print line. The text format must be ragged right and the print line must terminate with a carriage return. The print line length is determined by the value of the Line Length (default or specified). The print line consists of 1) preceding text, 2) space, and 3) subsequent text. For example:

CHAPTER 2<ESC>y2-1<CR>

will print out as:

CHAPTER 2 2-1

Only one <ESC>y command can be contained in a line of text (i.e., between Carriage Return commands).

FORCE ESCAPEMENT - <ESC>l(nn) (lowercase "ell")

This command allows the user to specify that the printhead is to be moved left or right the specified distance (in mils). If used between two printable ASCII characters, the space is in addition to the "side-bearing" space built into each character. The space may be either left or right from the current position.

When moving to the right, the distance must be in the range from 1 to 2047 mils. A movement to the left is obtained by specifying a negative value (two's complement) for the distance. When moving to the left, the command must be preceded by a printable character and cannot exceed the width of the preceding character. (Exception: if a Vertical Spacing Command immediately precedes the <ESC>l command, negative spacing, up to 2048 mils or the width of the previous line (discounting any indent setting), can be specified.)

NON-ESCAPE - <ESC>N

This command causes the printer to print the next character on top of the character just printed (i.e., centered one on top of the other). The only commands that may appear between the two printable characters are a Select Font (<ESC>a) command and a Non-Escape (<ESC>N) command (they must be entered in that order). This command is useful, for example, when it is desired to differentiate between an O (letter O) and a zero (0) by using a slash through one or the other. This is expressed by:

0<ESC>N/ which prints as Ø

This command works with both proportional and non-proportional fonts in any format.

Vertical Spacing Commands

The commands listed here are in addition to the Line Feed (<LF>) and Form Feed (<FF>) commands which are detailed in "Standard Printer Commands". All vertical spacing commands are executed at the point where they are inserted in the text and do not require a preceding Carriage Return (<CR>). Any text in the print buffer is printed before the command is executed.

HALF-LINE FEED - <ESC>U

The Half-Line Feed command causes the paper to advance one-half the default or specified line feed height.

NEGATIVE LINE FEED - <ESC><LF>

This command causes the printer to move the paper one vertical space (line feed) backward.

NEGATIVE HALF-LINE FEED - <ESC>D

This command causes the printer to move the paper one-half vertical space (line feed) backward.

FORCE LEADING (Mils) - <ESC>o(nn)

The Force Leading (mils) command causes the printer to space the paper in a vertical direction the number of 3.5 mil spaces specified by (nn). If the number is positive, the spacing is in the forward direction. If the number is negative, the spacing is backward.

FORCE LEADING (Inches) - <ESC>O(n)

The Force Leading (Inches) command works similar to the Force Leading (Mils) command except it causes the printer to space the number of 1/48 inch increments specified by (n). The value of (n) can be in the range from 0 to 63. Since 1/48 inch is not an even number of 3.5 mil steps, the system performs a rounding operation to approximate the distance and maintains a record of the remaining error that is used in subsequent rounding operations.

ABSOLUTE TAB TO LINE NUMBER - <ESC>v(nn)

This command causes the printer to advance the paper to a specific line. The number of that line is specified by the value (nn). The top of form (i.e., the first available print line) is defined as line number 0. A line is defined as the distance from the baseline of one line to the baseline of the next line (i.e., Line Height). This command is useful in maintaining a specific vertical location for particular data (e.g., salutations, chapter heads, page numbers, etc.). This command is used to advance to a line past the current line; it cannot move the paper backwards.

Text Format Commands

The text format commands are used to alter system default values or default values contained within each font.

SELECT FORMAT - <ESC>n(a)

This command selects the format of the text to be printed. The default format is flush left (ragged right). The values that can be specified for (a) are given below.

Format No.	(a)	Description
0	@	Flush left (ragged right)
1	A	Justified (flush left and right)
2	B	Justified with interletter spacing
3	C	Flush right (ragged left)
4	D	Centered (ragged left and right)

Formats 1 and 2 represent the two methods that the MEDIA 12/7 printer can justify copy. When format 1 is specified, the MEDIA 12/7 printer fits as many complete words as possible onto a print line and then adjusts the space between the words so that the print line is completely filled.

Format 2 works in a similar fashion, except that, in addition to interword spacing adustments being made, interletter spacing is adjusted using the maximum, optimum, and minimum spacing values associated with the current font (default or specified).

NOTES:

1. Non-proportional spaced (monospaced) fonts cannot be letter-spaced justified. That is, if Format 2 is specified for a non-proportional spacing font, the printed results are unpredictable.

2. When identical text is printed using flush left, flush right, and centered format modes, the resulting printed copy will, line-for-line, contain identical text. When the same text is printed in one of the justifying formats, an additional word may be included in each line.

3. The justifying formats will compress, as well as expand the space between words and letters, if applicable, using the values specified for the minimum and maximum word and letter spacing for the font. (See NOTE 4.) Default word and letter spacing for each font is given in Appendix C. The default walues may be overridden using the Set Letter Space (<ESC>j) and Set Word Space (<ESC>w) commands.

4. In Format 1, the system ignores the maximum interword spacing value and inserts sufficient space between the words to justify the line (any line, with two or more words, will be justified). In Format 2, the system uses the interword and interletter spacing values in effect and can print some lines that are not justified. It may be necessary to increase the maximum values for interletter and interword spacing to eliminate this possibility.

5. In Format 1, interletter spacing is zero (i.e., the only space between letters is the normal side-bearing space of each letter. In Format 2, the system uses the minimum and maximum values in effect.

6. When a line is printed using Format 2 and requires expansion, the system adds twice as much space to the interword spacing as it does to the interletter spacing. In like manner, when compressing a line, the system reduces the interword spacing twice as much as it reduces interletter spacing.

7. Tabs cannot be used in Formats 1 and 2 to produce well-aligned columns (use Format 0). Tabs can be used with Formats 1 and 2 to indent paragraphs.

SET LINE LENGTH - <ESC>e(nnn)

The Set Line Length command specifies the length of the print line in mils (0.001 inch). The line length is specified, in ASCII, by the value (nnn). The default print line length is 7000 mils (7 inches).

SET LEFT MARGIN - <ESC>f(nnn)

The Set Left Margin command specifies the distance from the left edge of the paper to the start of the print line. The distance is expressed in mils (0.001 inch) and has an acceptable range of 0 to 13000. The default value is one inch (1000 mils) and the print line will start within 8 mils of the specified point. This command, in conjunction with the Set Line Length command, determines the right margin. For example, the paper is 8 1/2 inches (8500 mils) wide, the left margin in 1 1/4 inches (1250 mils) and the print line is 6 3/4 inches (6750 mils): the right margin, therefore, is 1/2 inch (500 mils). To restore the default left margin, enter the command with a value of 0.

The Set Left Margin command should be established for a document and not changed due to its inter-relationship with the line length. To block text at a different left margin, it is best to use the Set Indentation command. This command alters the print line length (therefore, the right margin is maintained).

SET TOP MARGIN - <ESC>T(nnn)

The Set Top Margin command is used to establish the distance between the top of the page (or form) and the top of the first print line. The distance is specified in mils and the default value is 500.

When using cut sheets (manual or automatic sheet feeder), the top margin cannot be less than 425 mils due to the normal distance between the top of the sheet (when it is engaged by the friction roller) and the print head.

When using a tractor feed unit with continuous forms, or a roll paper feeder with roll paper, the top margin can be set to a minimum of 1 mil (zero resets the default value).

NOTE: When feeding cut sheets manually, do not insert a sheet until the PAPER OUT light illuminates. When the PAPER OUT light illuminates, insert the sheet and press START. (This is necessary for the printer to sense the difference between sheet and continuous forms feeding and to process them accordingly.)

SET BOTTOM MARGIN - <ESC>B(nnn)

The Set Bottom Margin command is used to establish the minimum distance from the last print line on a page and the bottom of that page. The distance is specified in mils and the default value is 500.

When the printer detects that the next line is to be printed within the specified bottom margin, it perfoms a form feed before printing the line. Vertical motion commands that cause the paper to be moved into the bottom margin will likewise cause the printer to perform a form feed. If the vertical motion command that caused the form feed is followed by a second vertical motion command, the second command will be executed prior to printing the next line of text.

A minimum of one print line must be left between the top margin and the bottom margin. Unspecified results may occur if the margins overlap.

One use of the Set Bottom Margin command is to assure that a paragraph heading does not appear as the last line on a page. To accomplish this, specify two Set Bottom Margin commands before each paragraph head, the first specifying the area in which a paragraph head should not appear and the second re-establishing the normal bottom margin.

Example: <ESC>B@<P<ESC>B@GTPARAGRAPH HEAD

instructs the printer to eject the sheet if "PARAGRAPH HEAD" were to fall within a two inch bottom margin. If the two inch margin is not violated, re-establish the bottom margin to 1/2 inch.

VERTICAL MARGIN DISABLE/ENABLE - <ESC>V(n)

The Vertical Margin Disable/Enable command is used to disable the top and bottom margins. The top and bottom margins each have a default value of 1/2 inch. The size of the margins can be changed (see above) but cannot be made less than 1 mil. Therefore, unless the margins are disabled, the system will automatically perform a form feed whenever the next line is to be printed within the bottom margin. This command is for use in those applications where the host determines page endings or where data is to be printed across page boundaries.

When using cut sheets, the top margin cannot be less than 425 mils due to the normal distance between the top of the sheet and the print head.

To disable the Top and bottom margins, specify <ESC>V@. To re-enable the vertical margins, specify <ESC>VA

SET SINGLE LINE INDENT - <ESC>I(nnn)

The Set Single Line Indent command allows the current line only to be indented. The indentation can be negative (in the left margin) or positive. The sum of the values specified for left margin, single line indent, and hanging indent cannot be less than 0 nor greater than the line length. The single line indent is in addition to the hanging indent (if specified).

For example: The hanging indent is set at 750 mils.

1. A single line indent of 1000 mils is specified. Text will start at 1750 mils to the right of the left margin.
2. A single line indent of -500 causes the text to start at 250 mils to the right of the left margin.
3. In a similar manner, a single line indent of -1000 causes the text to start 250 mils to the left of the left margin.

SET HANGING INDENT - <ESC>g(nnn)

This command causes the following print lines to be indented (i.e., increases, or decreases, the distance from the left edge of the paper to the start of the print line). In effect, the right margin is maintained and the print line is shortened or lengthened. The indentation is specified in mils and is the distance the indented print line is to be offset from the left margin. As with the single line indent, the sum of the values specified for left margin, single line indent, and hanging indent cannot be less than 0 nor greater than the line length. To cancel an indentation, specify a value of 0 in the Set Indentation command.

SET LINE HEIGHT - <ESC>k(nn)

The Set Line Height command is used to change the default line height (leading) for the current font. A Set Line Height command can be issued for each font incorporated in the printer. Leading is the distance from the baseline of one print line to the baseline of the next print line. The specified Line Height is retained until the leading is changed or the printer is initialized or RESET. Line heights are specified in increments of line feed steps (3.5 mils).

SET COLUMN WIDTH - <ESC>C(nnn)

The column width used in tab operations is normally the same width as the space character of the font in use when the tab is specified. (Space character widths are given in Table 8-3.) If font changes are made during the creation of a table, unwanted results may occur. To create a column width that will be used for all tabbing operations, no matter what font is in use, specify the desired column width, in mils, in place of (nnn). For example, for a column width of 120 mils, specify <ESC>C@Ax

SET LETTER SPACE - <ESC>j(nn).(nn).

This command overrides the default values for minimum and maximum interletter spacing for a single font. The minimum and maximum values establish a range from which the printer chooses the appropriate value on a line-by-line basis. Letter spacing is used in conjunction only with Format 2 (justified with letter and word spacing). The override values are in effect for a font until the interletter spacing is again changed. The spacing is measured in increments of one mil. The first value specified in the command is the minimum spacing. The minimum value must be negative and the magnitude must be less than the width of the narrowest charcter in the font. If a positive number is specified, the printer firmware will convert it into a negative number of the same magnitude. The maximum value must be equal to or greater than 1 and less than a maximum of 127 mils. The default for a particular spacing is restored by specifying a value of zero. See Table 8-2.

To change only one value, specify only that value and include the punctuation marks (comma and period) in the proper position. For example, to change only the maximum value, specify <ESC>j,(nn). To change only the minimum value, specify <ESC>j(nn),. The optimum value for interletter spacing between any two characters is built into the font and is derived from the side bearing of those characters.

SET WORD SPACE - <ESC>w(nn).(nn).(nn).

The Set Word Space command overrides the default values for minimum, optimum, and maximum interword spacing for a single font. The default values are set when the font is initially placed in use. Optimum word space is used during the printing of non-justified text. The range specified by the minimum and maximum values is used when justified text is to be printed.

The override values are in effect for the font until the interword spacing is again changed or the printer is initialized or RESET. The first value specified is the minimum spacing, the second is the optimum spacing, and the third is the maximum spacing. The values must be in the range from 0 to 2047 and must be in increasing size (i.e., minimum - optimum - maximum). The default for a particular spacing is restored by specifying a value of zero.

To change only one value, specify only that value and include the punctuation marks (commas and period) in the proper position. For example, to change only the minimum value, specify <ESC>w(nn),,. See Table 8-2.

Table 8-2 Word Space/Letter Space Usage

Format	Range	
	Word Spacing	Letter Spacing
0, 3, 4	Optimum only*	0
1	Minimum up to as much as is required to fill the line	0
2	Minimum to maximum	Minimum to maximum

* Can be altered using the Set Word Space command

SET FORM LENGTH - <ESC>I(nnn)

This command defines the length of the paper to be used. The length is specified in increments of mils (0.001 inch) and must be in the range of 0 to 32767. The default value for the form length is 11 inches (11000 mils) and, if changed, can be restored by specifying a value of zero in this command. Specifying the proper page length is important so that, when the Form Feed (FF) command is issued, the paper is moved through the printer to the appropriate end of page.

Tab Commands

The Horizontal Tab command is detailed under "Standard Printer Commands" at the beginning of this section.

TAB SET LIST - <ESC>p(nn),(nn),....,(nn).
TAB CLEAR LIST - <ESC>q(nn),(nn),....,(nn).

These two commands allow the setting and clearing of one or more tab stops by column number. The columns start at the left margin. A column has the same width as the Space character of the font in use, or the width specified in a Column Width command, therefore, a tab setting of @J is 10 columns to the right of the left margin. Table C-1 lists the width of the Space character for the various fonts.

CLEAR ALL HORIZONTAL TABS - <ESC>2

This command clears all horizontal tabs set using the Tab Set List command.

ABSOLUTE HORIZONTAL TAB TO COLUMN - <ESC>h(nn)

This command causes the printhead to be moved to a specific column on the print line. The columns start at the left margin. A column has the same width as the Space character of the font in use, or the width specified in the Column Width command, therefore, a tab setting of @M is 13 spaces to the right of the left margin or indent. If the requested column has been passed, this command is ignored.

Font Commands

The font commands are used to select a font from those installed in the system and to reassign font numbers, if desired.

SELECT FONT - <ESC>a(n)

The Select Font command is used to choose the font to be used to print the subsequent text. When the MEDIA 12/7 printer is turned on, initialized, or RESET, it scans the fonts that are installed and assigns consecutive numbers to them based on their physical location in the printer. The numbers (n) are from 0 to 15 (refer to Table B-3 for the appropriate ASCII character to use for (n)). Font 0 is the default font (i.e., it is the font that is used when no other font is selected or when a font is selected that is not installed in the printer.

Example: To select font 3, the proper command is <ESC>aC (C is the ASCII equivalent of 3).

ASSIGN LOGICAL FONT - <ESC>d(n)font number

This command is used to assign "logical" numbers to the fonts installed in the MEDIA 12/7. These numbers override the numbers assigned by the printer (as described in the Select Font command, above) and are in effect until the printer is initialized or RESET. Once logical fonts are assigned, they can be chosen for use by the Select Font command. The logical number to be assigned to the font is (n) where (n) is in the range from 0 to 15 (refer to Table B-3 for the appropriate ASCII character to use for (n)). The font number is the eight-character font identification number listed in Appendix C.

Example: To assign Helvesan 8pt Italic as font 4, the proper command sequence is <ESC>dDX5020508.

Miscellaneous Commands

ROUNDED LINE FEED - <ESC>K(n)

Each font contains a default line height (leading) value. For most standard fonts this value is either 47 or 35 steps (each step is 3.5 mils). This leading value is an approximation of 6 (or 8) lines per inch. This yields a total distance for 6 lines of 987 mils and for 8 lines of 980 mils. Software has been incorporated so that any given line is actually printed within one step (3.5 mils) of 6 or 8 lines per inch.

To disable the rounded line feed feature, specify <ESC>KA. With the rounded line feed feature disabled, the default leading value in the font is used. To re-enable the rounded line feed feature, specify <ESC>K@. (Note: This feature works only with font whose default line heights are 35 or 47 steps. For all other fonts, the leading value is used.)

TOGGLE DRAFT MODE - <ESC>t

This command causes subsequent print lines to be printed in the draft equivalents of the specified non-draft fonts. It performs the same function as the DRAFT pushbutton on the MEDIA 12/7 Operator's Control Panel. When it first receives the <ESC>t command, the MEDIA 12/7 substitutes the equivalent draft font for the current font. As other standard fonts are selected, the MEDIA 12/7 printer substitutes their equivalent draft fonts (if any). When a second <ESC>t command is received, the printer stops substituting draft font equivalents and uses the selected fonts.

This command is in effect for a minimum of one complete print line. If the first and second <ESC>t commands are received as part of the same print line, the second cancels the first and there is no font change. Draft and standard fonts cannot be mixed on a single print line using this command. Mixing of fonts on a print line can be accomplished using the Select Font (<ESC>a) command.

BOLDING - <ESC>b(n)

The Bolding command is used to delimit a block of text that is to be bolded. Bolding is accomplished by printing the block of text specified a second time using the amount of horizontal displacement specified by (n). If the value specified for (n) reflects that bit 5 (the highest) is set, bolding will start and the displacement will be the value specified in the remaining 5 bits. If bit 5 is off, bolding will stop. When bolding is turned off, all remaining bits should also be off (i.e., specify <ESC>b@). (See "Printer Control Language" at the start of this section.) (A 4 mil displacement was used to bold the major headings in this manual.)

For example, to bold a block of text with a 4 mil offset, use the Bolding command as follows:

<ESC>bdThe time is now, said the _<ESC>b@

The time is now, said the _

(d specifies a 4 mil offset (a=1, b=2, _, f=6, etc.) and @ indicates that all bits are off.)

INITIALIZE PRINTER - <ESC><SUB>I

This command causes the internal logic of the printer to be reset. The conditions that exist after initialization are listed in Table 8-1. The command is a capital I (eye), not a lower case l (ell).

SET RIBBON USAGE - <ESC>r(n)

The Set Ribbon Usage command specifies the amount of ribbon to be used during one pass of the printhead from right to left. (The ribbon is advanced only when the printhead is traveling in the same direction as ribbon movement - advanced once for every two lines of a one-pass font, once for every line of a two-pass font, and twice for every line of a four-pass font.) The value specified for (n) is the number of 0.22 inch increments the ribbon is to be moved (the default value moves the ribbon 0.44 inch). The values for (n) can be in the range from 0 to 13. If the value 0 is specified, the default rate of ribbon usage is reinstated.

This command allows the user to establish a compromise between ribbon usage and print quality/copy density. For example, with tabular copy, ribbon usage can be set at a lower rate than is required for narrative text because of the lower copy density of tabular material.

Ribbon usage can be estimated by dividing the length of the ribbon by the usage rate. For example, if the usage rate is 7 (1.54 inches) and the ribbon length is 900 feet, approximately 14000 lines of a one-pass font, 7000 lines of a two-pass font, or 3500 lines of a four-pass font can be printed.

Table B-3. Decimal/ASCII Conversion

n	n	n	ASCII Character	n	n	ASCII Character	n	n	ASCII Character
0	0	0	@	1408	22	V	2816	44	l
4096	64	1	A	1472	23	W	2880	45	m
8192	128	2	B	1536	24	X	2944	46	n
12288	192	3	C	1600	25	Y	3008	47	o
16384	256	4	D	1664	26	Z	3072	48	p
20480	320	5	E	1728	27	[	3136	49	q
24576	384	6	F	1792	28	\	3200	50	r
28672	448	7	G	1856	29	]	3264	51	s
32768	512	8	H	1920	30	^	3328	52	t
36864	576	9	I	1984	31	_	3392	53	u
40960	640	10	J	2048	32	`	3456	54	v
45056	704	11	K	2112	33	a	3520	55	w
49152	768	12	L	2176	34	b	3584	56	x
53248	832	13	M	2240	35	c	3648	57	y
57344	896	14	N	2304	36	d	3712	58	z
61440	960	15	O	2368	37	e	3776	59	{
	1024	16	P	2432	38	f	3840	60	|
	1088	17	Q	2496	39	g	3904	61	}
	1152	18	R	2560	40	h	3968	62	~
	1216	19	S	2624	41	i	4032	63	<DEL>
	1280	20	T	2688	42	j			
	1344	21	U	2752	43	k			

NEGATIVE NUMBER CONVERSION

To convert a negative number to its ASCII equivalent, subtract the magnitude of the number (unsigned value) from 64, 4096, or 65536 (for one, two, and three ASCII characters respectively).

Table B-1. Ranges of Numbers/ASCII Representation

Number of ASCII Characters	Range
(n)	-32 to 31
(nn)	-2048 to 2047
(nnn)	-32768 to 32767

ANNEXE 5: QUELQUES COMMANDES DU VT100

digital

EK-VT100-RC-001

VT100 PROGRAMMING REFERENCE CARD

CURSOR CONTROL KEY CODES

Cursor Key (arrow)	VT52 Mode	ANSI/Cursor Key Mode Reset	ANSI/Cursor Key Mode Set
Up	ESC A	ESC [A	ESC O A
Down	ESC B	ESC [B	ESC O B
Right	ESC C	ESC [C	ESC O C
Left	ESC D	ESC [D	ESC O D

SPECIAL GRAPHICS CHARACTERS

Octal Code	Graphic with US or UK Set	Graphic with "Special Graphics" Set
137	—	Blank
140	\	♦ Diamond
141	a	▒ Checkerboard (error indicator)
142	b	HT horizontal tab
143	c	FF form feed
144	d	CR carriage return
145	e	LF line feed
146	f	° Degree symbol
147	g	± Plus/minus
150	h	NL new line
151	i	VT vertical tab
152	j	┘ Lower-right corner
153	k	┐ Upper-right corner
154	l	┌ Upper-left corner
155	m	└ Lower-left corner
156	n	+ Crossing lines
157	o	– Horizontal line - Scan 1
160	p	– Horizontal line - Scan 3
161	q	– Horizontal line - Scan 5
162	r	– Horizontal line - Scan 7
163	s	– Horizontal line - Scan 9
164	t	├ Left "T"
165	u	┤ Right "T"
166	v	┴ Bottom "T"
167	w	┬ Top "T"
170	x	| Vertical Bar
171	y	≤ Less than or equal to
172	z	≥ Greater than or equal to
173	{	π Pi
174	|	≠ Not equal to
175	}	£ UK pound sign
76	~	· Centered dot

ANSI COMPATIBLE MODE

CURSOR MOVEMENT COMMANDS

Cursor up	ESC [Pn A
Cursor down	ESC [Pn B
Cursor forward (right)	ESC [Pn C
Cursor backward (left)	ESC [Pn D
Direct cursor addressing	ESC [Pl; Pc H
Direct cursor addressing	ESC [Pl; Pc f
Index	ESC D
Next Line	ESC E
Reverse index	ESC M
Save cursor and attributes	ESC 7
Restore cursor and attributes	ESC 8

LINE SIZE (DOUBLE-HEIGHT AND DOUBLE-WIDTH) COMMANDS

Change this line to double-height top half	ESC # 3
Change this line to double-height bottom half	ESC # 4
Change this line to single-width single-height	ESC # 5
Change this line to double-width single-height	ESC # 6

CHARACTER ATTRIBUTES

ESC [Ps;Ps;Ps;....Ps m

Ps =		
	0 or None	All Attributes Off
	1	Bold on
	4	Underscore on
	5	Blink on
	7	Reverse video on

ERASING

From cursor to end of line	ESC [K
From cursor to end of line	ESC [0 K
From beginning of line to cursor	ESC [1 K
Entire line containing cursor	ESC [2 K
From cursor to end of screen	ESC [J
From cursor to end of screen	ESC [0 J
From beginning of screen to cursor	ESC [1 J
Entire screen	ESC [2 J

PROGRAMMABLE LEDs

ESC [Ps;Ps;...Ps q

Ps =		
	0 or None	All LEDs Off
	1	L1 on
	2	L2 on
	3	L3 on
	4	L4 on

CHARACTER SETS (G0 AND G1 DESIGNATORS)

Character Set	G0 Designator	G1 Designator
United Kingdom (UK)	ESC (A	ESC) A
United States (USASCII)	ESC (B	ESC) B
Special graphics characters and line drawing set	ESC (0	ESC) 0
Alternate character ROM	ESC (1	ESC) 1
Alternate character ROM special graphics characters	ESC (2	ESC) 2

SCROLLING REGION

ESC [Pt ; Pb r

TAB STOPS

Set tab at current column	ESC H
Clear tab at current column	ESC [g
Clear tab at current column	ESC [0 g
Clear all tabs	ESC [3 g

MODES

	To Set		To Reset	
Mode Name	**Mode**	**Sequence**	**Mode**	**Sequence**
Line feed/new line	New line	ESC [20h	Line feed	ESC [20/ *
Cursor key mode	Application	ESC [?1h	Cursor	ESC [?1/ *
ANSI/VT52 mode	ANSI		VT52	ESC [?2/ *
Column mode	132 Col	ESC [?3h	80 Col	ESC [?3/ *
Scrolling mode	Smooth	ESC [?4h	Jump	ESC [?4/ *
Screen mode	Reverse	ESC [?5h	Normal	ESC [?5/ *
Origin mode	Relative	ESC [?6h	Absolute	ESC [?6/ *
Wraparound	On	ESC [?7h	Off	ESC [?7/ *
Auto repeat	On	ESC [?8h	Off	ESC [?8/ *
Interlace	On	ESC [?9h	Off	ESC [?9/ *
Graphic proc option	On	ESC 1	Off	ESC 2
Keypad mode	Application	ESC =	Numeric	ESC >

*The last character of the sequence is a lowercase L (154_8).

REPORTS

Cursor Position Report

Invoked by	ESC [6 n
Response is	ESC [Pl; Pc R

Status Report

Invoked by	ESC [c
Invoked by	ESC [0 c
Response is	ESC [?1 ;Ps c

What Are You

Invoked by	ESC [c
Invoked by	ESC [0 c
Response is	ESC [?1 ;Ps c

Ps =	
0	Base VT100, no options
1	Processor option (STP)
2	Advanced video option (AVO)
3	AVO and STP
4	Graphics processor option (GPO)
5	GPO and STP
6	GPO and AVO
7	GPO, STP, and AVO

Alternately invoked by ESC Z (not recommended). Response is the same.

RESET

ESC c

CONFIDENCE TESTS

Fill Screen with "Es"	ESC # 8
Invoke Test(s)	ESC [2 ; Ps y

Ps =	
1	Power-up self test (ROM checksum, RAM, NVR, keyboard and AVO if installed)
2 (Loop back connector required)	Data Loop Back
4 (Loop back connector required)	EIA Modem Control Test
8	Repeat selected test(s) indefinitely (until failure or power off)

VT52 COMPATIBLE MODE

Cursor Up	ESC A	
Cursor Down	ESC B	
Cursor Right	ESC C	
Cursor Left	ESC D	
Select Special Graphics character set	ESC F	
Select ASCII character set	ESC G	
Cursor to home	ESC H	
Reverse line feed	ESC I	
Erase to end of screen	ESC J	
Erase to end of line	ESC K	
Direct cursor address	ESC Pl Pc	(see note 1)
Identify	ESC Z	(see note 2)
Enter alternate keypad mode	ESC =	
Exit alternate keypad mode	ESC >	
Enter ANSI mode	ESC <	

NOTE 1: Line and column numbers for direct cursor address are single character codes whose values are the desired number plus 37_8. Line and column numbers start at 1.

NOTE 2: Response to ESC Z is ESC / Z.

AUXILIARY KEYPAD CODES

Key	VT52 Numeric Mode	VT52 Application Mode	ANSI Numeric Mode	ANSI Application Mode
0	0	ESC?p	0	ESC O p
1	1	ESC ? q	1	ESC O q
2	2	ESC ? r	2	ESC O r
3	3	ESC ? s	3	ESC O s
4	4	ESC ? t	4	ESC O t
5	5	ESC ? u	5	ESC O u
6	6	ESC ? v	6	ESC O v
7	7	ESC ? w	7	ESC O w
8	8	ESC ? x	8	ESC O x
9	9	ESC ? y	9	ESC O y
– (minus)	– (minus)	ESC ? m	– (minus)	ESC O m
, (comma)	, (comma)	ESC ? / *	, (comma)	ESC O / *
. (period)	. (period)	ESC ? n	. (period)	ESC O n
ENTER	Same as RETURN	ESC ? M	Same as RETURN	ESC O M
PF1	ESC P	ESC P	ESC O P	ESC O P
PF2	ESC Q	ESC Q	ESC O Q	ESC O Q
PF3	ESC R	ESC R	ESC O R	ESC O R
PF4	ESC S	ESC S	ESC O S	ESC O S

*The last character of the sequence is a lowercase L (154_8).

7-BIT ASCII CODE

Octal Code	Char	Octal Code	Char	Octal Code	Char	Octal Code	Char
000	NUL	040	SP	100	@	140	`
001	SOH	041	!	101	A	141	a
002	STX	042	"	102	B	142	b
003	ETX	043	#	103	C	143	c
004	EOT	044	$	104	D	144	d
005	ENQ	045	%	105	E	145	e
006	ACK	046	&	106	F	146	f
007	BEL	047	' (apos)	107	G	147	g
010	BS	050	(	110	H	150	h
011	HT	051	)	111	I	151	i
012	LF	052	*	112	J	152	j
013	VT	053	+	113	K	153	k
014	FF	054	, (comma)	114	L	154	l
015	CR	055	– (minus)	115	M	155	m
016	SO	056	. (period)	116	N	156	n
017	SI	057	/	117	O	157	o
020	DLE	060	0	120	P	160	p
021	DC1	061	1	121	Q	161	q
022	DC2	062	2	122	R	162	r
023	DC3	063	3	123	S	163	s
024	DC4	064	4	124	T	164	t
025	NAK	065	5	125	U	165	u
026	SYN	066	6	126	V	166	v
027	ETB	067	7	127	W	167	w
030	CAN	070	8	130	X	170	x
031	EM	071	9	131	Y	171	y
032	SUB	072	:	132	Z	172	z
033	ESC	073	;	133	[	173	{
034	FS	074	<	134	\	174	|
035	GS	075	=	135	]	175	}
036	RS	076	>	136	^	176	~
037	US	077	?	137	_	177	DEL

NOTE: The following control characters are generated differently from previous DIGITAL terminals.

Code	VT100	Previous Terminal
NUL	CTRL – Space bar	CTRL – @
RS	CTRL – ~	CTRL – ^
US	CTRL – ?	CTRL – _

ANNEXE 6: LES COMMANDES DE DEPLACEMENT DE NEQN.

Les différentes commandes de NEQN sont :

<ESC>7 : monter d'une ligne;
<ESC>8 : monter d'une demi-ligne;
<ESC>9 : descendre d'une demi-ligne;
<CTRLH> : reculer d'une caractère;
<CTRLN> caractère <CTRLO> : caractère du deuxième jeu.

La liste des caractères générés par NEQN correspond à un mot qui l'identifie :

Neqn Input	Neqn Output
>=	>\b_
<=	<\b_
!=	/\b=
+-	+\b_
==	=\b_
cdot	<ESC>8.<ESC>9
CDOT	<ESC>8.<ESC>9
times	x
TIMES	x
SIGMA	<^N>R<^O>
pi	<^N>J<^O>
PI	<^N>P<^O>
alpha	<^N>A<^O>
beta	<^N>B<^O>
gamma	<^N>\\e<^O>
GAMMA	<^N>G<^O>
delta	<^N>D<^O>
epsilon	<^N>S<^O>
omega	<^N>C<^O>
DELTA	<^N>W<^O>
LAMBDA	<^N>E<^O>
PHI	<^N>F<^O>
OMEGA	<^N>Z<^O>
lambda	<^N>L<^O>
mu	<^N>M<^O>
nu	<^N>@<^O>
theta	<^N>T<^O>
rho	<^N>K<^O>
sigma	<^N>Y<^O>
tau	<^N>I<^O>
phi	<^N>U<^O>
INF	<^N>o<^O>
INFINITY	<^N>o<^O>
inf	<^N>o<^O>
infinity	<^N>o<^O>
partial	<^N>]<^O>
PARTIAL	<^N>]<^O>

zeta	<^N>Q<^O>
eta	<^N>N<^O>
iota	i
kappa	k
xi	<^N>X<^O>
omicron	o
upsilon	u
chi	X
psi	<^N>V<^O>
THETA	<^N>O<^O>
XI	X
UPSILON	U
PSI	<^N>H<^O>
del	<^N>[<^O>
DEL	<^N>[<^O>
nothing	
NOTHING	
approx	~\b=
APPROX	~\b=

ANNEXE 7: REALISATION DETAILLEE DE BACK_SPACE,ECRAN,PREFILTRE

1. BACK_SPACE

1.1. LES DIFFERENTS MODULES

La découpe choisie pour réaliser les traitements de back_space est illustrée au chapitre 4,figure 3.Les différents modules intervenant dans celle-ci sont :

1. module principal
2. initialisation
3. trt_sanders
4. gdeparenth
5. accolade
6. dernier
7. pmorceau
8. premier
9. racinecarre
10. signespec
11. pgenersign
12. underscore
13. pbackspace
14. pescape
15. pnroff
16. cde4
17. cde3
18. cde2
19. ignore
20. copy_maj
21. dec_ascii
22. caractimpr
23. font_value

1.2. NOTATIONS

Dans la description des modules,nous prendrons en considération un certain nombre de conventions concernant les commandes de la Sanders :

<DV> : déplacement vertical du type <ESC>o(nn)
<DH> : déplacement horizontal du type <ESC>l(nn)
Cette commande est souvent entourée de commandes de déplacements verticaux qui s'annulent,pour une raison inhérente à la Sanders.
<BS> : déplacement horizontal négatif dont la valeur correspond à la distance parcourue par une suite de commandes <CTRLH>. est du type <ESC>l(nn).

De plus,certaines variables du module principal et trt_sanders se retrouvent presque systématiquement dans les

autres modules.Leur signification étant inchangée,elles seront spécifiées une fois pour toutes.

1.3. DETAILS DES DIFFERENTS MODULES DE BACK_SPACE

1.3.1. module principal

entrée : textin

sortie : textout

effet :

Ce module gère chaque ligne du fichier d'entrée afin qu'elle soit traitée si elle contient des commandes.Chaque ligne est alors recopiée dans le fichier de sortie.

pré-conditions : /

post-conditions :

le fichier textout contient uniquement des caractères imprimables et des commandes connues par la Sanders.

procédures appelantes : /

procédures appelées :

initialisation
trt_sanders

variables locales :

textin : fichier d'entrée
textout: fichier de sortie
in_put : tampon d'entrée et sa longueur
out_put: tampon de sortie et sa longueur
policecour : numéro logique de la police de caractères courante
font : matrice qui contient pour chaque police de caractères la largeur de tous les caractères ascii imprimables.
sanders : matrice qui contient pour tous les caractères ascii imprimables,la séquence de commandes Sanders pour générer tout caractère du deuxieme jeu de caractères de nroff et les caractères supplémentaires générés par prefiltre.
action : tableau qui contient le choix du traitement à effectuer pour chaque caractère ascii.
escape : tableau qui contient,pour tous les caractères ascii "car",les spécifications nécessaires quand on rencontre une commande du type <ESC>car...
-le chiffre des centaines C =
0 si la commande n'est pas connue
entier positif sinon
-le chiffre des dizaines D =
1 si la listes des arguments de la commande termine par un point
0 sinon
-le chiffre des unités U =

nombre d'arguments de la commande si C≠0 et D=0
quelconque sinon
test : position du dernier <CTRLH> dans la ligne lue dans textin
pascde : booléen
true si il y a des commandes dans la ligne lue
false sinon

algorithme logique :

appel initialisation
copier une ligne du fichier d'entree dans in_put et
vérifier simultanément s'il y a des commandes à traiter.
si commande à traiter,appel de trt_sanders
copie du tampon dans le fichier de sortie

1.3.2. initialisation

entrées : /

sorties : /

effet :

Cette procédure initialise les variables globales policecour,font,sanders, action et escape.

pré-conditions : /

post-conditions : /

procédure appelante : module principal

procédures appelées : /

variables globales :

policecour
font
sanders
action
escape

1.3.3. trt_sanders

entrées : in-put

sorties : out_put

effet :

Cette procédure considère chaque caractère de in_put afin de sélectionner le traitement à effectuer.Le résultat du

traitement se trouve dans out_put.

pré-conditions : /

post-condition :

out_put contient uniquement des caractères imprimables et des commandes Sanders.

procédure appelante : module principal

procédures appelées :

copy_maj
pnroff
pescape
pbackspace
underscore
signespec
racinecarre
gdeparenth
ignore

variables globales : action,policecour

variables locales :

inn,out : position courante dans le tampon d'entrée,de sortie
index : numéro de l'action selon le caractère traité
identif : tableau utilisé lors du traitement des ‹CTRLH›
= 0 si ‹CTRLH›
= -entier si caractère à ignorer
= numéro logique de la police si caractère imprimable

algorithme logique :

tant que pas fin de in_put
choix de l'action à effectuer selon le caractère considéré de l'in_put
selon ce choix,sélectionner le traitement concerné

1.3.4. gdeparenth

entrées : in-put,out_put,identif,out,inn

sorties : in_put,out_put,identif,out,inn

effet :

Cette procédure gère,suivant les cas,la génération d'un changement de police
du passage aux caractères gras et réciproquement.
de grands crochets,accolades,barres,parenthèses.

pré-conditions :

- on considère dans in_put la séquence <CTRLO>car<CTRLH><CTRLN> où car vaut a,b,c,d,e,f,(,),[,],{,} ou |
- si car = (,),[,],{,} ou | ,la séquence suivante vaut
  <ESC>9..<ESC>9 |<CTRLH><ESC>8<ESC>8..|<CTRLH><ESC>8<ESC>8
  n-1 fois n fois
  <ESC>9..<ESC>9
  n+1 fois
- out est positionné à la première place libre de out_put.

post-conditions :

inn est positionné au prochain caractère à considérer dans in_put
identif est mis à jour
out est positionné à la prochaine place libre de out_put

procédure appelante : trt_sanders

procédures appelées :

copy_maj
ignore
cde3
accolade

variable globale : policecour

variable locale :

compt : contient le nombre de séquences |<CTRLH><ESC>8<ESC>8

algorithme logique :

```
suivant car,
     générer caractère gras
     générer fin caractère gras
     générer draft et mettre à jour policecour
     générer italique et mettre à jour policecour
     générer helvesan regular et mettre à jour policecour
     générer messenger (caractères initiaux)
             et mettre à jour policecour
     gérer les grandes accolades,...
             compter le nombre de <ESC>9 et les recopier
             mettre à jour identif
             générer grande accolade,...
```

1.3.5. accolade

entrées :

in-put,out_put,identif,out,inn,compt,car1,car2,car3 où

compt : nombre de séquences |<CTRLH><ESC>8

car1,car2,car3 : caractères pour générer la partie inférieure ,milieu,supérieure de l'accolade,...

sorties : in_put,out_put,identif,out,inn

effet :

Cette procédure gère la génération,de bas en haut,d'une grande accolade,crochet,parenthèse,barre.Il y a trois parties à gérer : les deux parties extrêmes et les parties intermédiaires.

pré-conditions :

compt : entier positif > 1
contient plus ou moins la moitié du nombre de morceaux à mettre dans la construction
out est positionné à la prochaine place libre de out_put

post-conditions :

inn est positionné au prochain caractère à considérer dans in_put
identif est mis à jour
out est positionné à la prochaine place libre de out_put

procédure appelante : gdeparenth

procédures appelées :

premier
pmorceau
dernier

variables globales : /

variables locales : /

algorithme logique :

génération de la partie inférieure (car1)
compt-2 appels de génération de partie milieu
appel de génération de la partie du milieu (car2)
compt-1 appels de génération de partie milieu
appel de génération de la partie supérieure (car3)

1.3.6. dernier

entrées :

in-put,out_put,identif,out,inn,car3
où car3 : correspond à un coin supérieur

sorties : in_put,out_put,identif,out,inn

effet :

Cette procédure génère la partie supérieure d'une grande accolade,.. c-a-d elle permet d'imprimer un coin carré ou arrondi et prend soin de souder ce coin au trait vertical précédemment imprimé, grâce à des commandes de déplacement verticaux.

pré-conditions :

car3 est le caractère ascii correspondant au coin supérieur dans le jeu de caractère scientifique.
out est positionné à la première place libre dans out_put.

post-conditions :

inn est positionné au prochain caractère à considérer dans in_put
out est positionné à la prochaine place libre de out_put

procédure appelante : accolade

procédures appelées :

cde2
cde3
cde4

variables globales : /

variables locales : /

algorithme logique :

<DV>
copier car3 dans out_put
<BS>
<ESC>8
<DV> (retour position courante)

1.3.7. pmorceau

entrées :

in-put,out_put,identif,out,inn,car2
où car2 correspond à une partie milieu d'une accolade

sorties : in_put,out_put,identif,out,inn

effet :

Cette procédure génère une partie intermédiaire d'une grande accolade,..

pré-conditions :

car2 est le caractère ascii correspondant au caractère milieu de l'accolade,... dans le jeu de caractère scientifique.
out est positionné à la première place libre dans out_put.

post-conditions :

inn est positionné au prochain caractère à considérer dans in_put
out est positionné à la prochaine place libre de out_put

procédure appelante : accolade

procédures appelées :

cde2
cde4

variables globales : /

variables locales : /

algorithme logique :

copier car2 dans out_put
<BS>
<ESC>8

1.3.8. premier

entrées :

in-put,out_put,identif,out,inn,car1
où car1 correspond à un coin inférieur

sorties : in_put,out_put,identif,out,inn

effet :

Cette procédure génère la partie inférieure d'une grande accolade,.. c-a-d elle permet d'imprimer un coin carré ou arrondi et prend soin de souder ce coin au trait vertical ultérieurementment imprimé, grâce à des commandes de déplacement verticaux.

pré-conditions :

car1 est le caractère ascii correspondant au coin inférieur dans le jeu de caractère scientifique.
out est positionné à la première place libre dans out_put.

post-conditions :

inn est positionné au prochain caractère à considérer

dans in_put
out est positionnè à la prochaine place libre de out_put

procèdures appelantes : accolade

procèdures appelèes :

cde2
cde3
cde4

variables globales : /

variables locales : /

algorithme logique :

copier car1 dans out_put
<BS>
<DV>
<ESC>8

1.3.9. racinecarre

entrèes : in-put,out_put,identif,out,inn

sorties : in_put,out_put,identif,out,inn

effet :

Cette procèdure remplace la construction de la racine carrèe donnèe par neqn :

\|<CTRLH><ESC>8..|<CTRLH><ESC>8 <ESC>8<ESC>8 _..._

par une nouvelle sèquence.

\| est remplacè par \<DV><BS>| pour combler le vide entre les deux caractères.

Chaque sèquence <CTRLH><ESC>8| est remplacèe par <BS><ESC>8|.

Les sèquences <ESC>8<ESC>8 du haut de l'intervalle sont remplacèes par <BS><ESC>8<DV>caractère <ESC>aCD<ESC>a@ <DV><BS><ESC>8 pour joindre le trait vertical et le trait horizontal
c-a-d on ajoute un coin et on ajuste horizontalement et verticalement afi d'avoir le haut de la racine d'un seul trait.

recopier les "_"

mettre à jour la position verticale

prè-conditions :

les deux caractères considèrès dans in_put sont \|

out est positionnè à la première place libre de out_put

post-conditions :

inn est positionnè au prochain caractère à considérer dans in_put
identif est mis à jour
out est positionnè à la prochaine place libre de out_put

procédure appelante : trt_sanders

procédures appelées :

cde2
cde3
cde4
ignore
pbackspace
copy_maj
underscore

variables globales : /

variables locales : /

algorithme logique :

gérer le remplacement de \|
tant que <CTRLH><ESC>8| gérer le remplacement
quand <ESC>8<ESC>8 gérer le remplacement
quand _..._ les recopier dans out_put
<DV> pour réajuster la hauteur
mettre à jour identif

1.3.10. signespec

entrées : in-put,out_put,identif,out,inn

sorties : in_put,out_put,identif,out,inn

effet :

Cette procédure gère le remplacement de signes contruits par nroff par le caractère correspondant dans le jeu scientifique.

prè-conditions :

le caractère considèrè dans in_put est <,>,=,/,+ ou ~.
out est positionnè à la première place libre de out_put.

post-conditions :

inn est positionnè au prochain caractère à considérer

dans in_put
identif est mis à jour
out est positionné à la prochaine place libre de out_put

procédure appelante : trt_sanders

procédures appelées : pgenersign

variables globales : /

variables locales : /

algorithme logique :

selon le signe considéré dans in_put
appeler pgenersign pour remplacer l'éventuelle construction.

1.3.11. pgenersign

entrées :

in-put,out_put,identif,out,inn,caract1,caract2
où caract1,caract2 : caractères

sorties : in_put,out_put,identif,out,inn

effet :

Cette procédure remplace le caractère construit par nroff du type car<BS>caract1 par le caractère correspondant ou recopie le caractère considéré si ce n'est pas une construction remplaçable.

pré-conditions :

le caractère considéré dans in_put est <,>,=,/ ,+ ou ~.
caract1 vaut "_" ou "="
caract2 est le caractère ascii correspondant au caractère voulu dans le jeu de caractères scientifiques.

post-conditions :

inn est positionné au prochain caractère à considérer dans in_put
identif est mis à jour
out est positionné à la prochaine place libre de out_put

procédure appelante : signespec

procédures appelées :

cde3
ignore
copy_maj

variables globales : /

variables locales : /

algorithme logique :

si car<CTRLH>caractl construction connue
copier le caractère correspondant et
mettre à jour identif
sinon recopier le caractère inchangé et
mettre à jour identif

1.3.12. underscore

entrées :

in-put,out_put,identif,out,inn,font_use
où font_use : numéro logique de la police courante

sorties : in_put,out_put,identif,out,inn

effet :

Quand,dans le jeu de caractères,"_" correspond bien à ce caractère là (ce n'est pas le cas du jeu de caractères grecs et du jeu de caractères scientifiques),cette procédure ajoute au début et à la fin de la séquence de "_",des commandes de déplacement verticaux afin de les imprimer un peu plus bas.

pré-conditions :

le caractère considéré dans in_put est "_".
out est positionné à la première place libre de out_put.

post-conditions :

inn est positionné au prochain caractère à considérer dans in_put
identif est mis à jour
out est positionné à la prochaine place libre de out_put

procédures appelantes :

trt_sanders
racinecarre

procédures appelées :

cde4
copy_maj

variables globales : /

variables locales :

uns_count : compteur de "_"

algorithme logique :

compter les "_"
si jeu de caractères grecs ou scientifiques
alors copier les "_" dans out_put et mettre à jour identif
sinon <DV>
copier les "_" dans out_put et mettre à jour identif
<DV>

1.3.13. pbackspace

entrées :

in-put,out_put,identif,out,inn,valinit
où valinit : distance initiale dont il faudra que la tête d'impression recule.

sorties : in_put,out_put,identif,out,inn

effet :

Cette procédure remplace une séquence de n <CTRLH> par une séquence Sanders de déplacement horizontal.Ce déplacement correspond à la distance utilisée pat les n caractères imprimables qui précèdent la séquence de <CTRLH>.

pré-conditions :

le caractère considéré dans in_put est <CTRLH>.

post-conditions :

inn est positionné au prochain caractère à considérer dans in_put
identif est mis à jour
out est positionné à la prochaine place libre de out_put

procédures appelantes :

trt_sanders
racinecarre

procédures appelées :

ignore
caractimpr
cde2
dec_ascii
la fonction font_value

variable globale : font

variables locales :

bs_count : compteur de <CTRLH>
bs_dist : distance dont il faudra reculer (en mils).

algorithme logique :

compter les <CTRLH> et mettre à jour identif
tant que bs_count non nul
rechercher le dernier caractère imprimable
soustraire à bs_dist la largeur de ce caractère
soustraire 1 de bs_count
convertir cette valeur entière en ascii et
mettre la commande de déplacement horizontal dans out_put

1.3.14. pescape

entrées : in-put,out_put,identif,out,inn

sorties : in_put,out_put,identif,out,inn

effet :

Cette procédure met à jour identif afin que cette commande ne soit pas prise en compte lors d'un traitement de backspaces.Pour certaines commandes Sanders particulières,elle effectue un traitement supplémentaire :

- <ESC>a(n) : changement de la valeur courante de policecour
- <ESC>N : on n'avance pas,il faut donc neutraliser le dernier caractère imprimable pour le traitement des backspaces.

pré-conditions :

le caractère considéré dans in_put est <ESC>.
out est positionné à la première place disponible de out_put

post-conditions :

inn est positionné au prochain caractère à considérer dans in_put
identif est mis à jour
out est positionné à la prochaine place libre de out_put

procédure appelante : trt_sanders

procédures appelées :

ignore
caractimpr
copy_maj

variables globales :

policecour
escape

variables locales : /

algorithme logique :

considérer la valeur du caractère suivant <ESC>dans le tableau escape
selon le chiffre des centaines
traiter commande inconnue
traiter <ESC>a(n) :
changer la valeur courante de policecour
copier la commande et mettre à jour identif
traiter <ESC>N
neutraliser le dernier caractère imprimable
copier la commande et mettre à jour identif
traiter une autre commande connue
copier la commande et mettre à jour identif

1.3.15. pnroff

entrées : in-put,out_put,identif,out,inn

sorties : in_put,out_put,identif,out,inn

effet :

Cette procédure remplace la séquence <CTRLN>caractère<CTRLO> par une séquence de commandes Sanders qui génère le caractère identifié.

pré-conditions :

la séquence de caractères considérée dans in_put est <CTRLN> caractère<CTRLO>.
out est positionné à la première place disponible de out_put

post-conditions :

inn est positionné au prochain caractère à considérer dans in_put
identif est mis à jour
out est positionné à la prochaine place libre de out_put

procédure appelante : trt_sanders

procédures appelées : /

variables globales : sanders

variables locales : /

algorithme logique :

calculer l'entrée dans la table sanders
copier cette ligne de la table dans out_put
mettre à jour identif

1.3.16. cde4

entrées :

in-put,out_put,identif,out,inn,nbreignore,caract1,caract2,caract3
où nbreignore : entier
caract1,caract2,caract3 : caractères

sorties : in_put,out_put,identif,out,inn

effet :

Cette procédure copie dans le tampon de sortie out_put,une commande ‹ESC› caract1 caract2 caract3 et met à jour nbreignore valeurs de identif.

pré-conditions :

out est positionné à la première place disponible de out_put

post-conditions :

out est positionné à la prochaine place libre de out_put
identif est mis à jour.
inn est positionné au prochain caractère à considérer dans in_put.

procédures appelantes :

underscore
racinecarre
premier
morceau
dernier

procédure appelée : ignore

variables globales : /

variables locales : /

algorithme logique :

copier la commande ‹ESC› caract1 caract2 caract3 dans out_put
nbreignore appels de ignore

1.3.17. cde3

entrées :

in-put,out_put,identif,out,inn,nbreignore,caract1,caract2
où nbreignore : entier
caract1,caract2 : caractères

sorties : in_put,out_put,identif,out,inn

effet :

Cette procédure copie dans le tampon de sortie out_put,une commande ‹ESC› caract1 caract2 et met à jour nbreignore valeurs de identif.

pré-condition :

out est positionné à la première place disponible de out_put

post-conditions :

out est positionné à la prochaine place libre de out_put
identif est mis à jour.
inn est positionné au prochain caractère à considérer dans in_put.

procédures appelantes :

pgenersign
gdeparenth
racinecarre
premier
dernier

procédure appelée : ignore

variables globales : /

variables locales : /

algorithme logique :

copier la commande ‹ESC› caract1 caract2 dans out_put
nbreignore appels de ignore

1.3.18. cde2

entrées :

in-put,out_put,identif,out,inn,nbreignore,caract1
où nbreignore : entier
caract1 : caractère

sorties : in_put,out_put,identif,out,inn

effet :

Cette procédure copie dans le tampon de sortie out_put,une commande <ESC> caractl et met à jour nbreignore valeurs de identif.

pré-condition :

out est positionné à la première place disponible de out_put

post-conditions :

out est positionné à la prochaine place libre de out_put
identif est mis à jour.
inn est positionné au prochain caractère à considérer dans in_put.

procédures appelantes :

racinecarre
pbackspace
premier
morceau
dernier

procédure appelée : ignore

variables globales : /

variables locales : /

algorithme logique :

copier la commande <ESC> caractl dans out_put
nbreignore appels de ignore

1.3.19. ignore

entrées :

identif,inn,font_use
où font_use : entier

sorties : identif,inn

effet :

Cette procédure met à la inn ème position de identif la valeur font_use et incrémente inn de 1.

pré-conditions : /

post-conditions : /

procédures appelantes :

trt_sanders
pescape
pbackspace
gdeparenth
pgenersign
cde2
cde3
cde4

procèdures appelèes : /

variables globales : /

variables locales : /

1.3.20. copy_maj

entrèes :

in-put,out_put,identif,out,inn,font_use,incr où
font_use,incr : entiers

sorties : in_put,out_put,identif,out,inn

effet :

Cette procèdure copie "incr" caractères de in_put à partir de inn dans out_put à partir de out.Il met à jour "incr" caractères de identif à partir de inn,avec la valeur font_use.

prè-conditions :

la sèquence des"incr" caractères considèrès dans in_put sont des caractères imprimables.
out est positionnè à la premiére place disponible de out_put

post-conditions :

inn est positionnè au prochain caractère à considèrer dans in_put
identif est mis à jour
out est positionnè à la prochaine place libre de out_put

procèdures appelantes :

trt_sanders
pescape
underscore
racinecarre
gdeparenth
pgenersign

procèdures appelèes : /

variables globales : /

variables locales : /

1.3.21. dec_ascii

entrées :

out_put,out,bs_dist,nbrearg
où bs_dist : distance à convertir en ascii
nbrearg : nombre de caractères ascii utilisés pour la conversion

sorties : out_put,out

effet :

Cette procédure convertit la distance décimale bs_dist en la valeur ascii correspondante pour la Sanders ("nbrearg" caractères).Il copie alors cette valeur dans out_put.

pré-conditions :

$1 \leq$ nbrearg ≤ 3
out est positionné à la première place disponible de out_put

post-condition :

out est positionné à la prochaine place libre de out_put

procédure appelante : pbackspace

procédures appelées : /

variables globales : /

variables locales :

n : table contenant 3 caractères et destiné à contenir le caractère ascii correspondant.

algorithme logique :

pour la distance bs_dist,
isoler les 6 bits de droite et le mettre dans un élément de la table.
conserver la valeur restant à considérer
si la distance est négative,considérer le 2's complement
convertir chaque élement de n en caractères
et les copier dans out_put

1.3.22. caractimpr

entrées :

identif,aux
où aux : position à partir de laquelle on recherche le dernier caractère imprimable

sortie : aux : position du dernier caractère imprimable

effet :

Etant donné une valeur d'indice "aux" de in_put,cette procédure fournit un indice inférieur ("aux" modifié) qui est la position du dernier caractère imprimable.

pré-conditions :

aux est positionné à la position suivant à partir de laquelle,il faut trouver le dernier caractère imprimable.
Ce caractère imprimable existe.

post-condition :

aux est positionné sur le caractère trouvé.

procédures appelantes :

pback_space
pescape

procédure appelée : caractimpr

variables globales : /

variables locales : /

algorithme logique :

```
tant dernier caractère imprimable pas trouvé
     si valeur de identif[aux] > 0 ,trouvé
     si valeur de identif[aux] < 0 ,soustraire 1 de aux
     si valeur de identif[aux] = 0 ,
             compter le nombre de <CTRLH>
             appel de ce nombre de fois caractimpr
             soustraire 1 de aux
```

1.3.23. font_value

entrée : i : entier

sortie : font_value : entier

effet :

Etant donnè i,la valeur logique d'une police de caractères,cette fonction donne la colonne du tableau font de cette police.

2. PREFILTRE

2.1. LES DIFFERENTS MODULES

La découpe choisie pour réaliser les traitements de prefiltre est donnée au chapitre 4 figure 4.Les différents modules intervenant dans celle-ci sont :

1. module principal
2. initialisation
3. pcas1
4. pcas2
5. pcas3
6. trtdelim
7. trtlignemath
8. pbold
9. pactioncle
10. pdelim
11. pleftright
12. cp_li_out
13. trtmacro
14. cp_li_input
15. pactionmacro
16. prebold
17. cp_input_buf
18. cp_buf_out
19. psearchword

2.2. CONVENTIONS

Les variables spécifiées dans le module principal se retrouvent dans de nombreux autres modules.Leur signification étant inchangée ,elles seront,sauf avis contraire,spécifiées une fois pour toutes.

2.3. DETAILS DES DIFFERENTS MODULES DE PREFILTRE

2.3.1. module principal

entrée : textin

sortie : textout

effet :

Ce module gère les modifications à apporter à des macros_instructions particulières ainsi qu'à quelques mots réservés,contenus dans le fichier textin.Le contenu de textin modifié se trouve dans textout.

pré-conditions : /

post-conditions : /

procédures appelantes : /

procédures appelées :

initialisation
pcas1
pcas2
pcas3

variables locales :

textin : fichier d'entrée
textout: fichier de sortie
in_put : tampon de travail pour une ligne de textin et sa longueur
out_put: tampon de travail pour placer un mot et sa longueur
macrotable :table contenant les macros qui nécessitent un traitement
table : table contenant des mots réservés qui nécessitent un traitement.
d1,d2 : délimiteurs
delimd1d2 : booléen à true si les expressions mathématiques sont délimitées par d1 et d2;à false sinon
delimEQEN : booléen à true si on est entre les macros-instructions .EQ et .EN ; à false,sinon.
bold_bool : booléen à true si partie de texte à écrire en caractères gras ; false,sinon.
font_bool : booléen à true si police particulière utilisée à false sinon.
delimiteur : ensemble des délimiteurs qui permettent d'isoler un mot dans une ligne.
nbre_bool : nombre de lignes à écrire en caractères gras.
nbre_font : nombre de lignes à écrire avec une police de caractères particlière.
longtable,longmacrotable : longueur de table et de macrotable

algorithme logique :

```
appel initialisation
tant que pas fin textin
     si on est entre les délimiteurs d1 et d2 :
             appel de pcas3
     si on est entre aucun délimiteur :
             appel de pcas1
     si on est entre les macros .EQ et .EN :
             appel de pcas2.
```

2.3.2. initialisation

entrées : /

sorties : /

effet :

Cette procédure initialise les tables des macros_instructions et des mots réservés concernés par un traitement.Elle met à false les variables booléenes.Elle initialise les longueurs des tables,elle met à " " d1 et d2 et définit l'ensemble des délimiteurs de mots.

pré-conditions : /

post-conditions : /

procédure appelante : module principal

procédures appelées : /

variables globales :

longtable,longmacrotable
delimd1d2,delimEQEN,bold_bool,font_bool
d1,d2
delimiteur
table
macrotable

2.3.3. pcas1

entrées :

textin,textout,in-put,tampon
delimEQEN,delimd1d2,bold_bool,font_bool
nbre_bool,nbre_font

sorties :

textin,textout
delimEQEN,delimd1d2,bold_bool,font_bool
nbre_bool,nbre_font

effet :

Cette procédure considère une ligne de textin.Si cette ligne contient une macro_instruction,elle la gère afin qu'elle soit éventuellement modifiée.Sinon,cette ligne est recopiée directement dans le fichier de sortie textout.Une gestion éventuelle de fin de caractère gras ou de fin de police est effectuée.

pré-conditions :

delimEQEN = false et delimd1d2 = false et pas fin fichier textin

post-conditions : /

procédure appelante : module principal

procédures appelées :

trtmacro
cp_li_input
cp_li_out
pbold

variables globales : /

variables locales : /

algorithme logique :

si le premier caractère de la ligne de textin est un "."
copier la macro_instruction dans in_put
traiter la macro-instruction
si le premier caractère de la ligne de textin n'est pas un "."
copier la ligne dans textout
traiter la fin de caractère gras
traiter la fin de police particulière

2.3.4. pcas2

entrées :

textin,textout,in-put,tampon,d1,d2
delimEQEN,delimd1d2,bold_bool,font_bool
nbre_bool,nbre_font

sorties :

textin,textout,d1,d2
delimEQEN,delimd1d2,bold_bool,font_bool
nbre_bool,nbre_font

effet :

Cette procédure copie une ligne de textin dans le tampon intermédiaire. Si la ligne considérée est une macro-instruction,elle sera gérée en vue d'une modification éventuelle.Sinon cette ligne sera gérée en tant que ligne contenant des expressions mathématiques et donc des mots réservés.

pré-conditions :

delimEQEN = true et pas fin de fichier textin

post-conditions : /

procédure appelante : module principal

procédures appelées :

trtmacro
cp_li_input
trtlignemath

variables globales : /

variables locales : /

algorithme logique :

copier la ligne de textin dans in_put
si macro-instruction,la traiter
sinon,traiter la ligne mathématique

2.3.5. pcas3

entrées :

textin,textout,in-put,tampon,d1,d2
delimEQEN,delimd1d2,bold_bool,font_bool
nbre_bool,nbre_font

sorties :

textin,textout,d1,d2
delimEQEN,delimd1d2,bold_bool,font_bool
nbre_bool,nbre_font

effet :

Cette procédure copie une ligne de textin dans le tampon intermédiaire. Si la ligne considérée est une macro-instruction,elle sera gérée en vue d'une modification éventuelle.Sinon cette ligne sera gérée en tant que ligne pouvant contenir des délimiteurs d1 et d2 et entre ceux-ci des mots réservés.

pré-conditions :

delimEQEN = false et delimd1d2 = true et
pas fin de fichier textin

post-conditions : /

procédure appelante : module principal

procédures appelées :

trtmacro
cp_li_input
trtdelim

variables globales : /

variables locales : /

algorithme logique :

copier la ligne de textin dans in_put
si macro-instruction,la traiter
sinon,traiter la ligne contenant d1 et d2

2.3.6. trtlignemath

entrées :

textin,textout,in-put,tampon,d1,d2,table,longtable
delimd1d2,bold_bool,font_bool
nbre_bool,nbre_font

sorties :

textin,textout,d1,d2,table,longtable
delimEQEN,delimd1d2,bold_bool,font_bool
nbre_bool,nbre_font

effet :

Cette procédure prend en charge la ligne contenue dans in_put.Elle gère son transfert dans textout en permettant des modifications éventuelles dues à la présence de certains mots réservés.Elle tient compte d'une fin de changement de police ou de caractères gras.

pré-conditions :

delimEQEN = true
Le premier caractère de in_put n'est pas un point.

post-conditions :

delimEQEN = true
La ligne de textin est traitée et écrite dans textout.

procédure appelante : pcas2

procédures appelées :

cp_input_buf
psearchword
pactioncle
cp_buf_out
pbold

variables globales : /

variables locales :

wordfound : true si un mot à traiter est trouvé
false,sinon
k : numéro de l'entrée de la table correspondant au mot réservé à traiter

algorithme logique :

```
tant que pas fin in_put
     copier un mot de in_put dans tampon
     rechercher si ce mot est dans table
     si wordfound gérer le traitement
     sinon copier le mot dans textout
traiter fin caractère gras
traiter fin police particulière
```

2.3.7. trtdelim

entrées :

textin,textout,in-put,tampon,d1,d2,table,longtable
delimd1d2,bold_bool,font_bool
nbre_bool,nbre_font

sorties :

textin,textout,d1,d2,table,longtable
delimEQEN,delimd1d2,bold_bool,font_bool
nbre_bool,nbre_font

effet :

Cette procédure prend en charge la ligne contenue dans in_put.Elle gère son transfert dans textout en permettant des modifications éventuelles dues à la présence de certains mots réservés,entre les délimiteurs d1 et d2 et en ne touchant pas aux parties extérieures à ceux-ci. elle tient compte d'une fin de changement de police ou de caractères gras.

pré-conditions :

delimEQEN = false et delimdld2 = true
Le premier caractère de in_put n'est pas un point.

post-conditions :

delimEQEN = true
La ligne de textin est traitée et écrite dans textout.

procédure appelante : pcas3

procédures appelées :

cp_input_buf
psearchword
pactioncle
cp_buf_out
pbold

variables globales : /

variables locales :

wordfound : true si un mot à traiter est trouvé
false,sinon
k : numéro de l'entrée de la table correspondant au mot
réservé à traiter

algorithme logique :

```
tant que pas fin in_put
     tant que pas d1,copier in_put dans textout
     quand d1 l'écrire dans textout
     tant que pas d2
             copier un mot de in_put dans tampon
             rechercher si ce mot est dans table
             si wordfound gérer le traitement
             sinon copier ce mot dans textout
     quand d2 l'écrire
traiter fin caractère gras
traiter fin police particulière
```

2.3.8. pbold

entrées :

bold_bool,nbre_bool,car2
où car2 : caractère

sorties : textout,bold_bool,nbre_bool

effet :

Cette procédure décompte le nombre de lignes restant à écrire dans une police particulière,ou en caractère gras,si c'est le cas. Si ce nombre décroissant atteint

l,elle gènère la commande identifiant la fin de modification d'écriture.

prè-conditions : /

post-conditions : /

procèdures appelantes :

pcasl
trtlignemath
trtdelim

procèdures appelèes : /

variables globales : /

variables locales : /

algorithme logique :

```
si bold_bool
     si fin traitement
             mettre bold_bool à false
             ecrire commande fin traitement
     sinon soustraire 1 du nombre de lignes "nbre_bool"
             restant à traiter
```

2.3.9. pactioncle

entrèes :

textout,in_put,tampon,table,d1,d2,k,i,delimd1d2
où k : position de l'entrèe dans la table
i : position du prochain caractère à considèrer dans in_put

sorties :

textout,in_put,tampon,table,d1,d2,delimd1d2

effet :

Cette procèdure sèlectionne l'action à effectuer pour un mot rèservè identifiè par la kème entrèe de la table des mots rèservès.

prè-conditions :

$1 \leq k \leq$ longtable
tampon contient un mot de table.

post-condition :

Le prochain mot à considèrer est dans in_put à partir de

la position i.

procédures appelantes :

trtlignemath
trtdelim

procédures appelées :

pleftright
pdelim

variables globales : /

variables locales : /

algorithme logique :

selon le numéro de l'action du kème élément de la table
sélectionner le traitement correspondant

2.3.10. pdelim

entrées :

textout,in_put,tampon,d1,d2,k,i,delimd1d2
où i : position du prochain caractère à considérer
dans in_put

sorties :

textout,in_put,tampon,d1,d2,delimd1d2

effet :

Cette procédure gère la définition de délimiteurs,ainsi que la fin de leur champ d'action.Si les délimiteurs sont en action,d1 contiendra le délimiteur de départ et d2 celui de fin.De plus, la variable booléenne delimd1d2 est mise à true.En cas de fin d'action,delimd1d2 est mis à false.

pré-conditions :

tampon contient "delim"
la séquence de caractères non blancs de in_put considérée, à partir de i,est "off" ou "car1 car2".

post-condition : delimd1d2 est mis à jour.

procédures appelantes :

trtlignemath
trtdelim

procédures appelées :

cp_buf_out
cp_input_buf

variables globales : /

variables locales : /

algorithme logique :

copier tampon dans textout
copier un mot de in_put dans tampon
si fin action délimiteurs
mettre à false delimd1d2
sinon mettre à true delimd1d2
mettre car1 dans d1
mettre car2 dans d2
copier tampon dans textout

2.3.11. pleftright

entrées :

textout,in_put,tampon,i
où i : position du caractère suivant à considérer dans in_put

sorties :

textout,in_put,i
où i : position du caractère suivant à considérer dans in_put

effet :

Cette procédure ajoute un identifiant du type <CTRLO>caractère <CTRLH><CTRLN> devant un mot réservé "left" ou "right". Le caractère dépend de ce qui suit le mot réservé.

pré-conditions :

tampon contient "left" ou "right"
et le caractère suivant non blanc à partir de i
vaut {,},|,[,],(ou).

post-conditions : /

procédure appelante : pactionclé

procédure appelée : cp_buf_out

variables globales : /

variable locale :

l : entier contenant la position dans tampon du caractère à mettre dans l'identifiant à insérer.

algorithme logique :

ajouter dans le tampon contenant le "left" ou le "right",le caractère suivant.
suivant la valeur de ce caractère,écrire dans textout, la séquence de caractères identifiant le contenu du tampon
copier tampon dans textout

2.3.12. cp_li_out

entrées :

textin,textout,ch
où ch : premier caractère de la ligne lue dans textin

sorties : textin,textout

effet :

Le premier caractère d'une ligne de textin est lu et donné par ch. Cette procédure copie la ligne de textin dans textout.

pré-conditions : /

post-conditions : /

procédure appelante : pcas1

procédures appelées : /

variables globales : /

variables locales : /

2.3.13. trtmacro

entrées :

textout,textin,in_put,tampon,macrotable,longtable
bold_bool,font_bool
nbre_bool,nbre_font

sorties :

textout,textin,macrotable,longtable
bold_bool,font_bool
nbre_bool,nbre_font

effet :

Cette procédure gère les macros-instructions rencontrées dans textin.Elle permet d'effectuer la transformation à opérer,s'il y a lieu. La ligne modifiée ou non est copiée dans textout.

pré-conditions :

le premier caractère de in_put est un ".".

post-condition : La ligne de textin est traitée et écrite dans textout.

procédures appelantes :

pcas1
pcas2
pcas3

procédures appelées :

cp_input_buf
psearchword
pactionmacro
cp_buf_out

variables globales : /

variables locales :

wordfound :true si macro_instruction à traiter;false sinon.
i : position courante de in_put
e : entrée dans la table si wordfound = true

algorithme logique :

```
tant que pas fin in_put
     copier un mot de in_put dans tampon
     chercher ce mot dans macrotable
     si macro-instruction à traiter,appel pactionmacro
     sinon copier le tampon dans textout
```

2.3.14. cp_li_input

entrées : textin,in_put,valinit

sorties : textin,in_put

effet :

Cette procédure copie la partie non lue de la ligne contenue dans textin,dans le tampon in_put à partir de la "valinit"ème place.

pré-condition : 1 ≤ valinit ≤ longueur maximale de in_put

post-conditions : /

procédures appelantes :

pcas1
pcas2
pcas3

procédures appelées : /

variables globales : /

variables locales : /

2.3.15. pactionmacro

entrées :

textout,tampon,macrotable
delimEQEN,bold_bool,font_bool
e,i,nbre_bool,nbre_font
où i : valeur courante du tampon in_put
e : position de l'entrée dans la macrotable

sorties :

textout,macrotable
delimEQEN,bold_bool,font_bool

effet :

Cette procédure sélectionne le traitement à effectuer pour la macro identifiée par la eème entrée de macrotable. ‹CTRLH›‹CTRLN› devant un mot réservé "left" ou "right". Le caractère dépend de ce qui suit le mot réservé.

pré-condition :

tampon contient une macro contenue dans macrotable à la eème position

post-condition :

La ligne contenant la macro-instruction est traitée.

procédure appelante : trtmacro

procédures appelées :

cp_buf_out
prebold

variables globales : /

variables locales : /

algorithme logique :

suivant la valeur de l'action correspondant à la eème entrée de macrotable,
gérer la macro_instruction correspondante

2.3.16. prebold

entrées :

textout,in_put,tampon,i,nbre_bool,bold_bool,car1
où i : valeur courante du tampon in_put
car1 : intervient dans la commande identifiante à ajouter

sortie :

textout,in_put,tampon,i,nbre_bool,bold_bool,car1

effet :

Cette procédure initialise un passage aux caractères gras ou à une autre police de caractères.

pré-condition :

tampon contient "BO","D","I" ou "HR"

post-conditions :

bold_bool et nbre_bool sont mis à jour.
La ligne contenant "BO","D","I" ou "HR" est totalement traitée.

procédure appelante : pactionmacro

procédure appelée : cp_input_buf

variables globales : /

variables locales : /

algorithme logique :

mettre bold_bool à true
écrire la commande identifiante de passage à un autre type d'écriture dans textout
copier un mot de in_put dans tampon
calculer du nombre de lignes dans ce type d'écriture et mettre ce nombre de lignes dans nbre_bool.

2.3.17. cp_input_buf

entrées :

in_put,i,d2
où i : position du caractère suivant à considérer dans in_put

sortie :

in_put,tampon,i,d2
où i : position du caractère suivant à considérer dans in_put

effet :

Cette procédure isole le mot de in_put situé à partir de la position i et le met dans tampon.

pré-conditions :

$1 \leq i \leq$ longueur du tampon
i est positionné sur un délimiteur ou sur le premier caractère du mot à copier dans tampon.

post-conditions :

Tampon contient le mot suivant de in_put.
i est positionné sur le caractère suivant à considérer dans in_put.

procédures appelantes :

trtmacro
trtdelim
trtlignemath

procédures appelées : /

variables globales : /

variables locales : /

algorithme logique :

tant qu'on a des délimiteurs,les écrire dans textout
tant qu'on n'a pas de délimiteurs,copier les caractères dans tampon
mettre à blanc la fin du tampon

2.3.18. cp_buf_out

entrées : textout,tampon

sortie : textout

effet :

Cette procédure copie le contenu de tampon dans le fichier de sortie textout.

pré-conditions : /

post-conditions : /

procédures appelantes :

trtmacro
trtlignemath
trtdelim
pactionmacro
pleftright
pdelim

procédures appelées : /

variables globales : /

variables locales : /

2.3.19. psearchword

entrées : tampon,table,longtable

sortie :

tampon,table,wordfound,e,longtable où wordfound : booléen
e : position de l'entrée de table si wordfound =true

effet :

Cette procédure compare le mot contenu dans tampon avec une table de longueur longtable.Si elle le trouve,wordfound est mis à true et e est la position de l'entrée dans la table.Sinon,wordfound est à false.

pré-conditions : /

post-conditions : /

procédures appelantes :

trtmacro
trtmacro
trtdelim

procédures appelées : /

variables globales : /

variables locales : /

algorithme logique :

```
initaliser e et wordfound
tant que mot pas trouvé et que pas fin table
     comparer le eème élément de la table au tampon
     si c'est égal ,alors mot trouvé
     sinon ,incrémenter e de 1
```

3. ECRAN

3.1. LES DIFFERENTS MODULES

La découpe choisie pour réaliser les traitements de écran est illustrée au chapitre 4,figure 5.Les différents modules intervenant dans celle-ci sont :

1. module principal
2. initialisation
3. trt_ecran
4. gdeparenth
5. accolade
6. dernier
7. pmorceau
8. premier
9. racinecarre
10. signespec
11. pgenersign
12. pbackspace
13. underscore
14. barrefract
15. pescape
16. pnroff
17. pmvt
18. pfont
19. cde2
20. ignore
21. copy_maj
22. ascii_dec

3.2. NOTATIONS

Certaines variables du module principal et trt_ecran se retrouvent presque systématiquement dans les autres modules.Leur signification étant inchangée,elles seront spécifiées une fois pour toutes.

3.3. DETAILS DES DIFFERENTS MODULES DE ECRAN

3.3.1. module principal

entrée : textin

sortie : textout

effet :

Ce module gère chaque ligne du fichier d'entrée afin qu'elle soit traitée si elle contient des commandes.Chaque ligne est alors recopiée dans le fichier de sortie.

pré-conditions : /

post-conditions :

le fichier textout contient uniquement des caractères imprimables et des commandes connues par le VT100.

procédures appelantes : /

procédures appelées :

initialisation
trt_ecran

variables locales :

textin : fichier d'entrée
textout: fichier de sortie
in_put : tampon d'entrée et sa longueur
out_put: tampon de sortie et sa longueur
video : matrice qui contient pour tous les caractères ascii imprimables,la séquence de commandes du VT100 pour générer tout caractère du deuxieme jeu de caractères de nroff et les caractères supplémentaires générés par prefiltre.
action : tableau qui contient le choix du traitement à effectuer pour chaque caractère ascii.
escape : tableau qui contient,pour tous les caractères ascii "car",les spécifications nécessaires quand on rencontre une commande du type ‹ESC›car...
-le chiffre des centaines C =
0 si la commande n'est pas connue
entier positif sinon
-le chiffre des dizaines D =
1 si la listes des arguments de la commande termine par un point
0 sinon
-le chiffre des unités U =
nombre d'arguments de la commande si C≠0 et D=0
quelconque sinon
nbre_li : compteur du nombre de lignes de textin

nbre_li_blan : compteur du nombre de lignes blanches consécutives
monter : booléen à true si la ligne lue contient des commandes
de déplacement vers le haut;à false sinon.
descendre : booléen à true si la ligne lue contient des commandes
de déplacement vers le bas;à false sinon.
bool_uns : booléen à true si la ligne est soulignée;à
false sinon.

algorithme logique :

appel initialisation
copier une ligne du fichier d'entree dans in_put
si ligne blanche
ajouter 1 à nbre_li_blan
mettre la longueur du tampon de sortie à 0
si pas ligne blanche
appel trt_ecran
si monter,écrire des lignes blanches supplémentaires
si descendre,enregistrer la demande de lignes blanches
supplémentaires
copie du out_put dans textout et à l'écran.

3.3.2. initialisation

entrées : /

sorties : /

effet :

Cette procédure initialise les variables globales video,
action et
escape,bool_uns,nbre_li,nbre_li_blan,monter,descendre.

pré-conditions : /

post-conditions : /

procédure appelante : module principal

procédures appelées : /

variables globales :

video
action
escape
bool_uns
nbre_li
nbre_li_blan
monter
descendre

3.3.3. trt_ecran

entrées : in-put,monter,descendre,bool_uns

sorties : out_put,monter,descendre,bool_uns

effet :

Cette procédure considère chaque caractère de in_put afin de sélectionner le traitement à effectuer.Le résultat du traitement se trouve dans out_put.

pré-conditions : /

post-condition :

out_put contient uniquement des caractères imprimables et des commandes du VT100.

procédure appelante : module principal

procédures appelées :

copy_maj
pnroff
pescape
pbackspace
signespec
racinecarre
gdeparenth
ignore
pmvt

variable globale : action

variables locales :

inn,out : position courante dans le tampon d'entrée,de sortie
index : numéro de l'action selon le caractère traité
identif : tableau utilisé lors du traitement des ‹CTRLH›
= 0 si ‹CTRLH›
= -1 si caractère à ignorer
= 1 si caractère imprimable

algorithme logique :

tant que pas fin de in_put
choix de l'action à effectuer selon le caractère considéré de l'in_put
selon ce choix,sélectionner le traitement concerné

3.3.4. gdeparenth

entrées : in-put,out_put,identif,out,inn,monter,descendre

sorties : in_put,out_put,identif,out,inn,monter,descendre

effet :

Cette procédure gère,suivant les cas,la génération
d'un changement de police
du passage aux caractères gras et réciproquement.
de grands crochets,accolades,barres,parenthèses.

pré-conditions :

- on considère dans in_put la séquence <CTRLO>car<CTRLH><CTRLN>
  où car vaut a,b,c,d,e,f,(,),[,],{,} ou |
- si car = (,),[,],{,} ou | ,la séquence suivante vaut
  <ESC>9..<ESC>9 |<CTRLH><ESC>8<ESC>8..|<CTRLH><ESC>8<ESC>8
  n-1 fois n fois
  <ESC>9..<ESC>9
  n+1 fois
- out est positionné à la première place libre de out_put.

post-conditions :

inn est positionné au prochain caractère à considérer
dans in_put
identif,monter et descendre sont mis à jour
out est positionné à la prochaine place libre de out_put

procédure appelante : trt_sanders

procédures appelées :

ignore
pmvt
accolade
cde2

variables globales : /

variable locale :

compt : contient le nombre de séquences |<CTRLH><ESC>8<ESC>8

algorithme logique :

suivant car,
    générer caractère gras
    générer fin caractère gras
    générer draft
    générer italique
    générer helvesan regular
    générer messenger (caractères initiaux)
    gérer les grandes accolades,...
        compter le nombre de <ESC>9 et les recopier
        mettre à jour identif,monter,descendre.
        générer grande accolade,...

3.3.5. accolade

entrées :

in-put,out_put,identif,out,inn,compt,car1,car2,car3 où

compt : nombre de séquences ö‹CTRLH›‹ESC›8
car1,car2,car3 : caractères pour générer la partie inférieure ,milieu,supérieure de l'accolade,...

sorties : in_put,out_put,identif,out,inn

effet :

Cette procédure gère la génération,de bas en haut,d'une grande accolade,crochet,parenthèse,barre.Il y a trois parties à gérer : les deux parties extrêmes et les parties intermédiaires.

pré-conditions :

compt : entier positif › 1
contient plus ou moins la moitié du nombre de morceaux à mettre dans la construction
out est positionné à la prochaine place libre de out_put

post-conditions :

inn est positionné au prochain caractère à considérer dans in_put
identif est mis à jour
out est positionné à la prochaine place libre de out_put

procédure appelante : gdeparenth

procédures appelées :

premier
pmorceau
dernier

variables globales : /

variables locales : /

algorithme logique :

génération de la partie inférieure (car1)
compt-2 appels de génération de partie milieu
appel de génération de la partie du milieu (car2)
compt-1 appels de génération de partie milieu
appel de génération de la partie supérieure (car3)

3.3.6. dernier

entrées :

in-put,out_put,identif,out,inn,car3
où car3 : correspond à un coin supérieur

sorties : in_put,out_put,identif,out,inn

effet :

Cette procédure génère la partie supérieure d'une grande accolade,.. c-a-d elle permet d'imprimer un coin carré .

pré-conditions :

car3 est le caractère ascii correspondant au coin supérieur dans le jeu de caractère scientifique.
out est positionné à la première place libre dans out_put.

post-conditions :

inn est positionné au prochain caractère à considérer dans in_put
out est positionné à la prochaine place libre de out_put

procédure appelante : accolade

procédures appelées :

pfont
pmvt

variables globales : /

variables locales : /

algorithme logique :

copier car3 dans out_put
monter d'une ligne
reculer le curseur d'un caractère

3.3.7. pmorceau

entrées :

in-put,out_put,identif,out,inn,car2
où car2 correspond à une partie milieu d'une accolade

sorties : in_put,out_put,identif,out,inn

effet :

Cette procédure génère une partie intermédiaire d'une

grande accolade,..

pré-conditions :

car2 est le caractère ascii correspondant au caractère milieu de l'accolade,... dans le jeu de caractère scientifique.
out est positionné à la première place libre dans out_put.

post-conditions :

inn est positionné au prochain caractère à considérer dans in_put
out est positionné à la prochaine place libre de out_put

procédure appelante : accolade

procédure appelée : pmvt

variables globales : /

variables locales : /

algorithme logique :

copier car2 dans out_put
monter d'une ligne
reculer d'un caractère

3.3.8. premier

entrées :

in-put,out_put,identif,out,inn,car1
où car1 correspond à un coin inférieur

sorties : in_put,out_put,identif,out,inn

effet :

Cette procédure génère la partie inférieure d'une grande accolade,.. c-a-d elle permet d'imprimer un coin carré.

pré-conditions :

car1 est le caractère ascii correspondant au coin inférieur dans le jeu de caractère scientifique.
out est positionné à la première place libre dans out_put.

post-conditions :

inn est positionné au prochain caractère à considérer dans in_put
out est positionné à la prochaine place libre de out_put

procèdures appelantes : accolade

procèdures appelèes :

pfont
pmvt

variables globales : /

variables locales : /

algorithme logique :

copier car1 dans out_put
monter d'une ligne
reculer d'un caractère

3.3.9. racinecarre

entrèes : in-put,out_put,identif,out,inn

sorties : in_put,out_put,identif,out,inn

effet :

Cette procèdure remplace la construction de la racine carrèe donnèe par neqn :

\|<CTRLH><ESC>8..|<CTRLH><ESC>8 <ESC>8<ESC>8 _..._

par une nouvelle sèquence.

\| est remplacè par \ <barre verticale graphique> .

Chaque sèquence <CTRLH><ESC>8| est remplacèe par <CTRLH><ESC>[A <barre verticale graphique>.

Les sèquences <ESC>8<ESC>8 du haut de l'intervalle sont remplacèes par <CTRLH><ESC>[1A<coin graphique> pour joindre le trait vertical et le trait horizontal .

remplacer chaque "_" par un <trait horizontal graphique>.

mettre à jour la position verticale

prè-conditions :

les deux caractères considèrès dans in_put sont \|
out est positionnè à la premiére place libre de out_put

post-conditions :

inn est positionnè au prochain caractère à considèrer dans in_put
identif est mis à jour

out est positionnè à la prochaine place libre de out_put

procèdure appelante : trt_sanders

procèdures appelèes :

pmvt
pfont
ignore

variables globales : /

variables locales : /

algorithme logique :

gèrer le remplacement de \|
tant que <CTRLH><ESC>8| gèrer le remplacement
quand <ESC>8<ESC>8 gèrer le remplacement
quand _..._ gèrer le remplacement
<DV> pour rèajuster la hauteur
mettre à jour identif

3.3.10. signespec

entrèes : in-put,out_put,identif,out,inn

sorties : in_put,out_put,identif,out,inn

effet :

Cette procèdure gère le remplacement de signes contruits par nroff par le caractère correspondant dans le jeu de caractères graphiques du VT100.

prè-conditions :

le caractère considèrè dans in_put est <,>,/ ou +.
out est positionnè à la première place libre de out_put.

post-conditions :

inn est positionnè au prochain caractère à considèrer dans in_put
identif est mis à jour
out est positionnè à la prochaine place libre de out_put

procèdure appelante : trt_ecran

procèdures appelèes : pgenersign

variables globales : /

variables locales : /

algorithme logique :

selon le signe considèrè dans in_put
appeler pgenersign pour remplacer l'éventuelle construction.

3.3.11. pgenersign

entrèes :

in-put,out_put,identif,out,inn,caract1,caract2
où caract1,caract2 : caractères

sorties : in_put,out_put,identif,out,inn

effet :

Cette procèdure remplace le caractère construit par nroff du type car<BS>caract1 par le caractère correspondant ou recopie le caractère considèrè si ce n'est pas une construction remplaçable.

prè-conditions :

le caractère considèrè dans in_put est <,>,/ ou +.
caract1 vaut "_" ou "="
caract2 est le caractère ascii correspondant au caractère voulu dans le jeu de caractères graphiques.

post-conditions :

inn est positionnè au prochain caractère à considèrer dans in_put
identif est mis à jour
out est positionnè à la prochaine place libre de out_put

procèdure appelante : signespec

procèdures appelèes :

pfont
ignore
copy_maj

variables globales : /

variables locales : /

algorithme logique :

si car<CTRLH>caract1 construction connue
copier le caractère correspondant et
mettre à jour identif
sinon recopier le caractère inchangè et
mettre à jour identif

3.3.12. pbackspace

entrées : in-put,out_put,identif,out,inn

sorties : in_put,out_put,identif,out,inn

effet :

Cette procédure gère une séquence de n ‹CTRLH› .Sa présence est dûe à une barre de fraction ou à un soulignement.

pré-conditions :

le caractère considéré dans in_put est ‹CTRLH›.

post-conditions :

inn est positionné au prochain caractère à considérer dans in_put
identif est mis à jour
out est positionné à la prochaine place libre de out_put

procédure appelante : trt_ecran

procédures appelées :

ignore
pescape
barrefract
underscore

variable globale : bool_uns

variables locales :

bs_count : compteur de ‹CTRLH›
uns_count : compteur de "_"
aux : position du premier ‹CTRLH›

algorithme logique :

```
compter les ‹CTRLH› et mettre à jour identif
tant que ‹ESC›9,les traiter
tant que "_",les compter et mettre à jour identif
si bs_count ≤ uns_count
     si ‹ESC›9 traiter barre fraction
     sinon traiter soulignement
si bs_count › uns_count,copier tout dans out_put
```

3.3.13. underscore

entrées :

in-put,out_put,identif,out,inn,aux,bs_count

où aux : position du premier <CTRLH>
bs_count : nombre de <CTRLH>

sorties : in_put,out_put,identif,out,inn

effet :

Cette procédure intercale des commandes de début et de fin de soulignement autour des bs_count caractères imprimables précédents les <CTRLH> de in_put (en évitant les commandes).

pré-conditions :

bs_count ≤ uns_count
la séquence considérée dans in_put à la position inn n'est pas <ESC>9.
out est positionné à la première place libre de out_put.

post-conditions :

inn est positionné au prochain caractère à considérer dans in_put
out est positionné à la prochaine place libre de out_put

procédure appelante : pbackspace

procédure appelée : pmvt

variables globales : /

variables locales :

extra : nombre de <CTRLH>
aux1 : position du dernier caractère considéré dans in_put

algorithme logique :

trouver la position du bs_count ème caractère imprimable
qui précède la séquence de <CTRLH> considérée.
si dernier caractère est un accent,englober le tout
insérer la commande de début de soulignement
copier la séquence de caractères dans out_put
insérer la commande de fin de soulignement

3.3.14. barrefract

entrées :

in-put,out_put,identif,out,inn,bs_count,uns_count
où bs_count : nombre de <CTRLH>
uns_count : nombre de "_"

sorties : in_put,out_put,identif,out,inn

effet :

Cette procédure trace une barre de fraction.

pré-conditions :

bs_count ≤ uns_count
la séquence considérée dans in_put à la position inn est <ESC>9.
out est positionné à la première place libre de out_put.

post-conditions :

inn est positionné au prochain caractère à considérer dans in_put
out est positionné à la prochaine place libre de out_put

procédure appelante : pbackspace

trt_ecran
racinecarre

procédures appelées :

cde2
pmvt
pfont

variables globales : /

variables locales : /

algorithme logique :

descendre d'une ligne
reculer de bs_count caractères
écrire uns_count traits horizontaux graphiques.

3.3.15. pescape

entrées :

in-put,out_put,identif,out,inn,monter,descendre,bool_uns

sorties :

in_put,out_put,identif,out,inn,monter,descendre,bool_uns

effet :

Cette procédure met à jour identif afin que cette commande ne soit pas prise en compte lors d'un traitement de backspaces.Elle traite et convertit en commandes du VT100 certaines commandes particulières de neqn et de

Sanders.
- les commandes de déplacement latéraux et verticaux de neqn et de Sanders sont converties (<ESC>N,2,U,9,7,D,8)
- la commande Sanders de soulignement est convertie (<ESC>u(nn)).
- de même,pour la commande de passage aux caractères gras (<ESC>b(n)).
- la commande <ESC>s(n)(a) est aussi convertie,les caractères à répéter sont recopiés.
- de même pour les commandes de tabulation absolue horizontale et verticale (<ESC>v(nn),<ESC>h(nn)).

pré-conditions :

le caractère considéré dans in_put est <ESC>.
out est positionné à la première place disponible de out_put

post-conditions :

inn est positionné au prochain caractère à considérer dans in_put
identif,monter,descendre,bool_uns sont mis à jour
out est positionné à la prochaine place libre de out_put

procédure appelante : trt_ecran

procédures appelées :

ignore
pmvt
cde2
asciidec

variable globale : escape

variables locales : /

algorithme logique :

considérer la valeur du caractère suivant <ESC> dans le tableau escape
selon le chiffre des centaines
    traiter commande inconnue
    traiter les commandes de déplacement verticaux et latéraux
        changer la valeur courante de monter ou descendre
        convertir la commande et mettre à jour identif
    traiter <ESC>u(n)
        convertir la commande et mettre à jour identif
        modifier la valeur de bool_uns
    traiter <ESC>h(nn)
        convertir la valeur ascii (nn) en décimal
        copier la commande convertie dans out_put et
        mettre à jour identif
    traiter de manière analogue <ESC>v(nn)
    traiter une autre commande connue
        mettre à jour identif

3.3.16. pnroff

entrées : in-put,out_put,identif,out,inn

sorties : in_put,out_put,identif,out,inn

effet :

Cette procédure remplace la séquence ‹CTRLN›caractère‹CTRLO› par une séquence de commandes du VT100 qui génère le caractère identifié ou un caractère différent muni d'un attribut visuel du VT100.

pré-conditions :

la séquence de caractères considérée dans in_put est ‹CTRLN› caractère‹CTRLO›.
out est positionné à la première place disponible de out_put

post-conditions :

inn est positionné au prochain caractère à considérer dans in_put
identif est mis à jour
out est positionné à la prochaine place libre de out_put

procédure appelante : trt_ecran

procédure appelée : ignore

variable globale : video

variables locales : /

algorithme logique :

calculer l'entrée dans la table sanders
copier cette ligne de la table dans out_put
mettre à jour identif

3.3.17. pmvt

entrées :

in-put,out_put,identif,out,inn,nbreignore,pn,lettre
où nbreignore,pn : entier
lettre : caractère

sorties : in_put,out_put,identif,out,inn

effet :

Cette procédure copie dans le tampon de sortie out_put,une commande ‹ESC› [‹nombre entier› ‹lettre› et met à jour nbreignore valeurs de identif.

pré-condition :

out est positionné à la première place disponible de out_put

post-condition :

out est positionné à la prochaine place libre de out_put

procédures appelantes :

underscore
racinecarre
premier
morceau
dernier
pescape
trtecran
gdeparenth

procédure appelée : ignore

variables globales : /

variables locales : /

algorithme logique :

copier la commande ‹ESC›[‹nombre entier› ‹lettre› dans out_put
nbreignore appels de ignore

3.3.18. pfont

entrées :

in-put,out_put,identif,out,inn,nbreignore,lettre
où nbreignore : entier
lettre : caractère

sorties : in_put,out_put,identif,out,inn

effet :

Cette procédure copie dans le tampon de sortie out_put,une commande ‹ESC› (lettre et met à jour nbreignore valeurs de identif.

pré-condition :

out est positionné à la première place disponible de out_put

post-condition :

out est positionné à la prochaine place libre de out_put

procédures appelantes :

pgenersign
barrefract
racinecarre
premier
dernier

procédure appelée : ignore

variables globales : /

variables locales : /

algorithme logique :

copier la commande <ESC> (lettre dans out_put
nbreignore appels de ignore

3.3.19. cde2

entrées :

in-put,out_put,identif,out,inn,nbreignore
où nbreignore : entier
lettre : caractère

sorties : in_put,out_put,identif,out,inn

effet :

Cette procédure copie dans le tampon de sortie out_put,une commande <ESC> lettre et met à jour nbreignore valeurs de identif.

pré-condition :

out est positionné à la première place disponible de out_put

post-condition :

out est positionné à la prochaine place libre de out_put

procédures appelantes :

accolade
gdeparenth
barrefract
racinecarre
pbackspace
premier
morceau
dernier

procédure appelée : ignore

variables globales : /

variables locales : /

algorithme logique :

copier la commande <ESC> lettre dans out_put
nbreignore appels de ignore

3.3.20. ignore

entrées :

identif,inn,font_use
où font_use : entier

sorties : identif,inn

effet :

Cette procédure met à la inn_ème position de identif la valeur font_use et incrémente inn de 1.

pré-conditions : /

post-conditions : /

procédures appelantes :

trt_ecran
pnroff
pescape
pbackspace
gdeparenth
pgenersign
cde2
pfont
underscore
gdeparenth

procédures appelées : /

variables globales : /

variables locales : /

3.3.21. copy_maj

entrées :

in-put,out_put,identif,out,inn,incr où incr : entier

sorties : in_put,out_put,identif,out,inn

effet :

Cette procédure copie "incr" caractères de in_put à partir de inn dans out_put à partir de out.Il met à jour "incr" caractères de identif à partir de inn,avec la valeur 1.

pré-conditions :

la séquence des"incr" caractères considérés dans in_put sont des caractères imprimables.
out est positionné à la première place disponible de out_put

post-conditions :

inn est positionné au prochain caractère à considérer dans in_put
identif est mis à jour
out est positionné à la prochaine place libre de out_put

procédures appelantes :

trt_ecran
pbackspace
pgenersign

procédures appelées : /

variables globales : /

variables locales : /

3.3.22. asciidec

entrées :

in_put,inn,nombre,nbrearg
où nombre : entier
nbrearg : nombre de caractères ascii utilisés pour la conversion

sorties : out_put,out

effet :

Cette procédure convertit les nbrearg caractères ascii de in_put dont le premier est à la position inn,en un nombre entier correspondant pour la Sanders (codage sixbit).Ces caractères font partie d'une commande Sanders.

pré-conditions :

$1 \leq$ nbrearg ≤ 3
les nbrearg caractères sont ceux d'une commande Sanders.
out est positionné à la première place disponible de out_put

post-condition :

out est positionnè à la prochaine place libre de out_put

procédure appelante : pescape

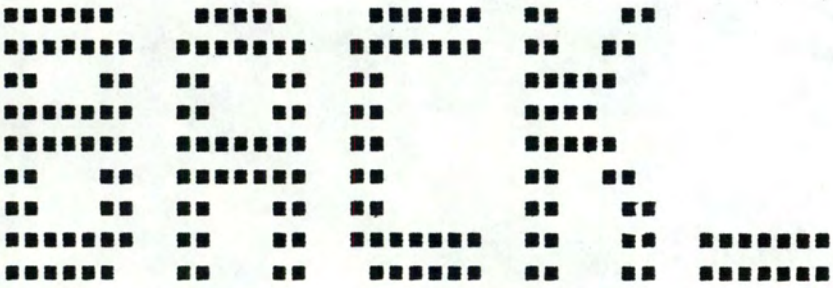
procédures appelées : /

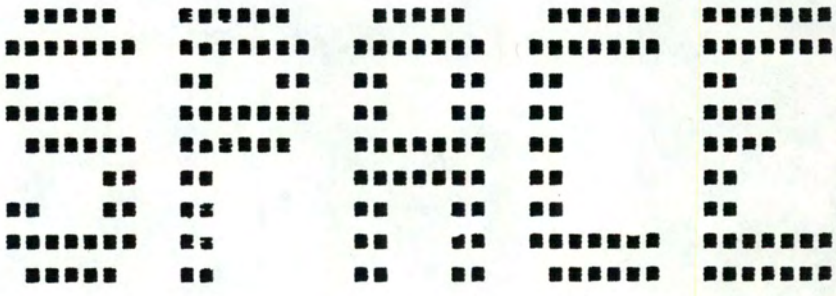
variables globales : /

variables locales : /

ANNEXE 8: PROGRAMMES

1. back_space
2. prefiltre
3. ecran





back_space.p

```
program back_space(textin, textout);

{cfr chapitre 4.8}

const BUFSIZE = 1500; {longueur maximale du tampon}
      NBRE_ARG_DFLT = 2;
      ASCII_COUNT = 95; {nombre de caracteres ascii imprimables}
      FONT_COUNT = 4; {nombre de polices dont la largeur des caracteres
                       different}
      SEQ_SIZE = 35; {longueur maximale de la sequence sanders correspondant
                      a un caractere du 2eme jeu decaracteres de nroff}
      TOTAL_ASCII = 128; {total des caracteres ascii}



type chainecar = packed array[1..BUFSIZE] of char; {representation du tampon}
     chainevar = record
                   caract : chainecar ; {tampon}
                   long   : integer ;   {longueur effective du tampon}
                   end;
     lignecar = packed array[1.. SEQ_SIZE] of char;
     lignevar = record
                  car    : lignecar ;
                  length : integer ;
                  end;
     matrice = packed array[1..ASCII_COUNT, 1..FONT_COUNT] of integer;
                  {contient la largeur des caracteres imprimables en mils
                   pour la police consideree}
     sanroff = array[1..ASCII_COUNT] of lignevar;
                  {contient la sequence sanders pour generer tout caractere
                   du 2eme jeu de nroff - en colonne 3, on a la lettre corres-
                   pondant au no de police a utiliser}
     selection = packed array[1..TOTAL_ASCII] of integer;
                  {- contient le numero du traitement a effectuer pour le
                     caractere donne.
                   - contient les specifications necessaires quand on
                     rencontre une commande escape.
                     chaque element est un entier de 3 chiffres :
                         - centaine = entier si cde connue ; 0 sinon
                         - dizaine  = 1 si la liste des arguments termine par 1
                                      point ; 0 sinon
                         - unite    = nombre d'arguments de la commande}
     chainentier = packed array[1..BUFSIZE] of integer; {utilise lorsque on
                    traite des backspaces. pour chaque caractere du tampon
                    d'entree, on a 1 valeur correspondante dans le tableau :
                         -entier positif : no de police
                         -zero           : backspace
                         -entier negatif : commande a ignorer (si on en
                          considere certaines on jouera sur la valeur de cet
                          entier)}


var textin, textout : text;
    in_put, out_put : chainevar;
    ch : char;
    i ,test, index, policecour : integer;
    pascde : boolean;
    font : matrice; {largeur caractere}
    sanders : sanroff ; {sequence generation sur sanders}
    action : selection; {choix du traitement pour un caractere}
    escape : selection; {specification pour la commande <esc>...}

{*****************************************************************}

                {PROCEDURE CARACTIMPR}

{*****************************************************************}

procedure caractimpr(var identif : chainentier; var aux : integer);
{cfr annexe 7.22 }

var trouve : boolean;
    compteur, i : integer;

begin
   trouve := false;
   aux := aux - 1;
   while (trouve = false) and (aux > 0) do
```

```
        begin
          if identif[aux] > 0
             then trouve := true {caract.imprimable dont le no de police est
                                  dans identif[aux]}
             else if identif[aux] < 0
                then aux := aux - 1 {commande a ignorer}
                else {sequence de <bs> : il faut les compter et sauter le
                      nombre correspondant de caracteres imprimables}
                  begin
                    compteur := 0;
                    repeat
                          compteur := compteur + 1;
                          aux := aux - 1;
                    until identif[aux] <> 0;
                    aux := aux + 1;
                    for i := 1 to compteur do caractimpr(identif,aux);
                    aux := aux - 1;
                  end;
        end;
end;

{*************************************************************************}

                {PROCEDURE DEC_ASCII}

{*************************************************************************}

procedure dec_ascii(var out_put : chainevar; var out,bs_dist : integer;
                    nbrearg : integer);
{cfr annexe 7.21 }

var nombre,i : integer;
    n : packed array[1..3] of integer;

begin
  for i := 1 to 3 do n[i] := 0;
  nombre := bs_dist;
  for i:=1 to nbrearg - 1 do
        begin
          n[4 - i] := nombre mod 64; {6 bits de droite isoles}
          nombre := nombre div 64; {bits qu'il reste a considerer}
        end;
  n[4 - nbrearg] := nombre;
  if bs_dist < 0 then
        begin
          n[1] := n[1] + 15;
          if n[2] <> 0 then n[2] := n[2] + 63
                       else begin
                            if bs_dist > -64 then n[2] := n[2] + 63
                                             else n[1] := n[1] + 1;
                            end;
          if n[3] <> 0 then n[3] := n[3] + 64
                       else n[2] := n[2] + 1;
        end;
  for i := 1 to nbrearg do
     out_put.caract[out + nbrearg - i] := chr(n[4 - i] + 64);
  out := out + nbrearg;
end;

{*************************************************************************}

                {FUNCTION FONT_VALUE}

{*************************************************************************}

function font_value(i : integer): integer;
{cfr annexe 7.22 }
{cette fonction associe a la "valeur logique" de la police,une valeur qui
 est la colonne a considerer dans la matrice font.pour l'instant on a :

        les differentes polices valeur dans identif     colonne de font
        ----------------------- ------------------      ---------------
 #0(@)messenger 12 germany      |        1              |   1 (12 pitch)
 #1(A)media max 12 germany      |        2              |   1
 #5(E)messenger 12 ascii        |        6              |   1
 #6(F)media max 12 ascii        |        7              |   1
 #7(G)helvesan regular ascii    |        8              |   3
 #4(D)helvesan italic ascii     |        5              |   4
 #3(C)scientific pi             |        4              |   2 (10 pitch)
 #2(B)greek mathematics         |        3              |   2
```

```
}

begin
  case i of
        1,2,6,7: font_value := 1;
              8: font_value := 3;
              5: font_value := 4;
          3,4: font_value := 2;
  end;
end;

{*************************************************************************}

                {PROCEDURE COPY_MAJ}

{*************************************************************************}

procedure copy_maj(var out_put,in_put : chainevar;var identif : chainentier;
                   var out,inn : integer; font_use,incr : integer);
{cfr annexe 7.19 }

var k : integer;

begin
  for k := 1 to incr do
        begin
          out_put.caract[out + k -1] := in_put.caract[inn + k -1];
          identif[inn +k -1] := font_use;
      end;
  inn := inn + incr;
  out := out + incr;
end;


{*************************************************************************}

                {PROCEDURE IGNORE}

{*************************************************************************}

procedure ignore(var identif : chainentier; var inn : integer;
                 font_use : integer);
{cfr annexe 7.19 }

begin
  identif[inn] := font_use;
  inn := inn +1;
end;

{*************************************************************************}

                {PROCEDURE CDE2}

{*************************************************************************}

procedure cde2(var out_put,in_put : chainevar;var identif : chainentier;
                   var out,inn : integer; nbreignore : integer; caract1 : char);
{cfr annexe 7.18 }
var i : integer;

begin
  out_put.caract[out] := chr(27);
  out_put.caract[out + 1] := caract1;
  out := out + 2;
  for i := 1 to nbreignore do ignore(identif,inn,-1);
end;

{*************************************************************************}

                {PROCEDURE CDE3}

{*************************************************************************}

procedure cde3(var out_put,in_put : chainevar;var identif : chainentier;
                   var out,inn : integer; nbreignore : integer;
                   caract1,caract2 : char);
{cfr annexe 7.18 }
var i : integer;

begin
  out_put.caract[out] := chr(27);
```

```
  out_put.caract[out + 1] := caract1;
  out_put.caract[out + 2] := caract2;
  out := out + 3;
  for i := 1 to nbreignore do ignore(identif,inn,-1);
end;

{*************************************************************************}

                {PROCEDURE CDE4}

{*************************************************************************}

procedure cde4(var out_put,in_put : chainevar;var identif : chainentier;
               var out,inn : integer; nbreignore : integer;
               caract1,caract2,caract3 : char);

{cfr annexe 7.17 }
var i : integer;

begin
  out_put.caract[out] := chr(27);
  out_put.caract[out + 1] := caract1;
  out_put.caract[out + 2] := caract2;
  out_put.caract[out + 3] := caract3;
  out := out + 4;
  for i := 1 to nbreignore do ignore(identif,inn,-1);
end;

{*************************************************************************}

                {PROCEDURE PNROFF}

{*************************************************************************}

procedure pnroff(var out_put,in_put : chainevar; var identif : chainentier;
                 var out,inn : integer);

{cfr annexe 7.16 }


var i,j : integer;

begin
  i := ord(in_put.caract[inn + 1]) - 31; {entree dans la table sanders}
  with sanders[i] do
    begin
      for j := 1 to length do out_put.caract[out + j -1] := car[j];
      out := out + length;
      identif[inn + 1] := ord(car[3])-63; {no police adequat}
    end;
  identif[inn] := -1;
  identif[inn + 2] := -1;
  inn := inn + 3;
end;

{*************************************************************************}

                {PROCEDURE PESCAPE}

{*************************************************************************}

procedure pescape (var out_put,in_put : chainevar; var identif : chainentier ;
                   var out,inn : integer);
{cfr annexe 7.15 }

var carcde,cde,dot,nbrearg,aux1:integer;

begin
  carcde := escape[ord(in_put.caract[inn + 1]) + 1]; {valeur contenue dans le
                                  tableau escape correspondant a la lettre
                                  suivant le <ESC>.}
  cde := carcde div 100; {quotient de la division par 100 : chiffre des cent.}
  case cde of
       0 : {commande inconnue : on ne recopie pas <esc> en output
                                on l'ignore si <bs> a traiter.}
           ignore(identif,inn,-1);
       1 : {commande connue}
           begin
             carcde := carcde mod 100; {reste : dizaine-unite}
             dot := carcde div 10; {chiffre des dizaines}
             if dot = 0 {cde dont la liste des arg. ne termine pas par "."}
                then begin
```

```
                        nbrearg := carcde mod 10; {unite = #arg}
                        copy_maj(out_put,in_put,identif,out,inn,-1,nbrearg+2)
                      end
                else begin
                        while in_put.caract[inn] <> '.' do
                        copy_maj(out_put,in_put,identif,out,inn,-1,1);
                        copy_maj(out_put,in_put,identif,out,inn,-1,1);
                      end;
              end;
        2 : {commande <ESC>a(n) : on la copie et on met la valeur identif a -1
             on modifie la valeur de policecour}
             begin
               policecour := ord(in_put.caract[inn + 2]) - 63;
               copy_maj(out_put,in_put,identif,out,inn,-1,3);
             end;
        3 : {commande <ESC>N : on n'avance pas,il faut neutraliser le dernier
             caractere imprimable pour le traitement des <BS>}
             begin
               aux1 := inn ;
               caractimpr(identif,aux1);
               identif[aux1] := -1;
               copy_maj(out_put,in_put,identif,out,inn,-1,2);
             end;
  end;
end;

{************************************************************************}

                {PROCEDURE PBACKSPACE}

{************************************************************************}

{cfr annexe 7.14 }
procedure pbackspace(var out_put,in_put : chainevar; var identif : chainentier
                     var out,inn : integer ;valinit : integer);



var bs_count,bs_dist,aux,i,j: integer;

begin
  bs_count := 1; {nombre de backspaces}
  bs_dist := valinit;  {distance dont il faudra que la tete d'impression
                        recule}
  aux := inn;
  ignore(identif,inn,0);

  while in_put.caract[inn] = chr(8) do {on compte les <BS>}
        begin
          bs_count := bs_count + 1;
          ignore(identif,inn,0);
        end;

  while bs_count <> 0 do        {traitement des <BS>}
        begin
          caractimpr(identif,aux); {dernier caractere imprimable}
          j := font_value(identif[aux]); {colonne du font}
          i := ord(in_put.caract[aux])-31; {ligne de font}
          bs_dist := bs_dist - font[i,j];
          bs_count := bs_count - 1;
        end;
  cde2(out_put,in_put,identif,out,inn,0,'U');
  out_put.caract[out] := chr(27);
  out_put.caract[out + 1] := '1';
  out :=out + 2;
  dec_ascii(out_put,out,bs_dist,2); {mise dans output de la valeur decimale
                                     convertie et incrementation de out}
  cde2(out_put,in_put,identif,out,inn,0,'D');
end;

{************************************************************************}

                {PROCEDURE UNDERSCORE}

{************************************************************************}

procedure underscore(var out_put,in_put : chainevar ;
                     var identif : chainentier ; var out,inn : integer ;
                     font_use : integer);
{cfr annexe 7.13 }
```

```
var uns_count : integer;

begin
  uns_count := 1;
  while in_put.caract[inn + uns_count] = '_'
        do uns_count := uns_count + 1;
  if (font_use = 3) or (font_use = 4)
        then copy_maj(out_put,in_put,identif,out,inn,font_use,uns_count)
        else begin
               cde4(out_put,in_put,identif,out,inn,0,'o','@','I');
               copy_maj(out_put,in_put,identif,out,inn,font_use,uns_count);
               cde4(out_put,in_put,identif,out,inn,0,'o',chr(127),'w');
             end;
end;

{************************************************************************}

                {PROCEDURE PGENERSIGN}

{************************************************************************}

procedure pgenersign(var out_put,input : chainevar;var identif : chainentier;
                     var out,inn : integer; caract1,caract2 : char);
{cfr annexe 7.12 }


begin
  if (in_put.caract[inn + 1] = chr(8)) and (in_put.caract[inn + 2] = caract1)
     then
        begin
          cde3(out_put,in_put,identif,out,inn,2,'a','C');
          out_put.caract[out] := caract2;
          out := out + 1;
          ignore(identif,inn,4); {jeu scientifique}
          cde3(out_put,in_put,identif,out,inn,0,'a','@');
        end
     else
        copy_maj(out_put,in_put,identif,out,inn,policecour,1);
end;



{************************************************************************}

                {PROCEDURE SIGNESPEC}

{************************************************************************}

procedure signespec(var out_put,in_put : chainevar; var identif : chainentier;
                    var out,inn : integer);
{cfr annexe 7.11 }
{Cette procedure est appelee par trtsanders lorsqu'on a eventuellement un
signe particulier a generer}

begin
  case in_put.caract[inn] of
         '>' : pgenersign(out_put,in_put,identif,out,inn,'_','h');
         '<' : pgenersign(out_put,in_put,identif,out,inn,'_','H');
         '=' : pgenersign(out_put,in_put,identif,out,inn,'_','N');
         '/' : pgenersign(out_put,in_put,identif,out,inn,'=','j');
         '+' : pgenersign(out_put,in_put,identif,out,inn,'_','L');
         '~' : pgenersign(out_put,in_put,identif,out,inn,'=','K');
  end;
end;

{************************************************************************}

                {PROCEDURE RACINECARRE}

{************************************************************************}

procedure racinecarre(var out_put,in_put : chainevar ;
                      var identif : chainentier ; var out,inn : integer);

{cfr annexe 7.10 }

begin
  out_put.caract[out] := in_put.caract[inn]; {\}
  out := out + 1;
  cde4(out_put,in_put,identif,out,inn,0,'o','@','G');{<ESC>o@G}
```

```
  cde2(out_put,in_put,identif,out,inn,0,'U');{<ESC>U<ESC>1<DEL>R<ESC>D}
  cde4(out_put,in_put,identif,out,inn,0,'1',chr(127),'R');
  cde2(out_put,in_put,identif,out,inn,0,'D');
  cde3(out_put,in_put,identif,out,inn,0,'a','C');{<ESC>aC}
  out_put.caract[out] := 'd';          {'d'}
  out := out + 1;
  cde3(out_put,in_put,identif,out,inn,0,'a','@');{<ESC>a@}
  ignore(identif,inn,policecour);{\}
  ignore(identif,inn,4);{dans jeu scientifique}

  while in_put.caract[inn] = chr(8) do              {^H<ESC>8!}
      begin
        pbackspace(out_put,in_put,identif,out,inn,0);    {^H}
        copy_maj(out_put,in_put,identif,out,inn,-1,2); {<ESC>8}
        cde3(out_put,in_put,identif,out,inn,0,'a','C');{<ESC>aC}
        out_put.caract[out] := 'd';          {'d'}
        out := out + 1;
        cde3(out_put,in_put,identif,out,inn,0,'a','@');{<ESC>a@}
        ignore(identif,inn,4);{4eme jeu}
      end;

  if (in_put.caract[inn] = chr(27)) and (in_put.caract[inn +1] = '8')
    then   {<ESC>8<ESC>8}
      begin
        cde2(out_put,in_put,identif,out,inn,0,'U');{^H = <ESC>U<ESC>1~\<ESC>D}
        cde4(out_put,in_put,identif,out,inn,0,'1','~','\');
        cde2(out_put,in_put,identif,out,inn,0,'D');
        copy_maj(out_put,in_put,identif,out,inn,-1,2); {<ESC>8}
        cde4(out_put,in_put,identif,out,inn,0,'o','@','I');{<ESC>o@I}
        cde3(out_put,in_put,identif,out,inn,0,'a','C');{<ESC>aC}
        out_put.caract[out] := 'D';          {'D'}
        out := out + 1;
        cde3(out_put,in_put,identif,out,inn,0,'a','@');{<ESC>a@}
        cde4(out_put,in_put,identif,out,inn,0,'o',chr(127),'t');{<ESC>o<DEL>t}
        cde2(out_put,in_put,identif,out,inn,0,'U');{^H=<ESC>U<ESC>1<DEL>v<ESC>
        cde4(out_put,in_put,identif,out,inn,0,'1',chr(127),'v');
        cde2(out_put,in_put,identif,out,inn,0,'D');
        copy_maj(out_put,in_put,identif,out,inn,-1,2); {<ESC>8}
      end;

  if in_put.caract[inn] = '_'      {_ _ _ _ _ _}
     then underscore(out_put,in_put,identif,out,inn,policecour);
  cde4(out_put,in_put,identif,out,inn,0,'o',chr(127),'!');{<ESC>o<DEL>!}

  if in_put.caract[inn] = chr(8)          {^H ^H ... ^H ^H}
    then pbackspace(out_put,in_put,identif,out,inn,54);
end;

{*************************************************************************}

                {PROCEDURE PREMIER}

{*************************************************************************}

procedure premier(var out_put,in_put : chainevar;
                  var identif : chainentier;
                  var out,inn : integer; car1 : char);
{cfr annexe 7.9 }
begin
  cde3(out_put,in_put,identif,out,inn,0,'a','C');{<ESC>aC}
  out_put.caract[out] := car1;   {car1}
  out := out + 1;
  cde4(out_put,in_put,identif,out,inn,0,'o','@','I');{<ESC>o@I}
  cde2(out_put,in_put,identif,out,inn,0,'U');{^H = <ESC>U<ESC>1~\<ESC>D}
  cde4(out_put,in_put,identif,out,inn,0,'1','~','\');
  cde2(out_put,in_put,identif,out,inn,0,'D');
  cde2(out_put,in_put,identif,out,inn,0,'8');{<ESC>8}
end;

{*************************************************************************}

                {PROCEDURE PMORCEAU}

{*************************************************************************}

procedure pmorceau(var out_put,in_put : chainevar;
                  var identif : chainentier;
                  var out,inn : integer; car2 : char);
{cfr annexe 7.8 }

begin
  out_put.caract[out] := car2;
```

```
  out := out + 1;
  cde2(out_put,in_put,identif,out,inn,0,'U');{^H = <ESC>U<ESC>1~\<ESC>D}
  cde4(out_put,in_put,identif,out,inn,0,'1','~','\');
  cde2(out_put,in_put,identif,out,inn,0,'D');
  cde2(out_put,in_put,identif,out,inn,0,'8');{<ESC>8}
end;

{*************************************************************************}

                {PROCEDURE DERNIER}

{*************************************************************************}

procedure dernier(var out_put,in_put : chainevar ;
                  var identif : chainentier;
                  var out,inn : integer; car3 : char);
{cfr annexe 7.7 }

begin
  cde4(out_put,in_put,identif,out,inn,0,'o','@','I');{<ESC>o@I}
  out_put.caract[out] := car3;
  out := out + 1;
  cde3(out_put,in_put,identif,out,inn,0,'a','@');{<ESC>a@}
  cde2(out_put,in_put,identif,out,inn,0,'U');{^H = <ESC>U<ESC>1~\<ESC>D}
  cde4(out_put,in_put,identif,out,inn,0,'1','~','\');
  cde2(out_put,in_put,identif,out,inn,0,'D');
  cde2(out_put,in_put,identif,out,inn,0,'8');{<ESC>8}
  cde4(out_put,in_put,identif,out,inn,0,'o',chr(127),'n');{<ESC>o<DEL>n}
end;

{*************************************************************************}

                {PROCEDURE ACCOLADE}

{*************************************************************************}

procedure accolade(var out_put,in_put : chainevar;
                   var identif : chainentier;
                   var out,inn,compt : integer;
                   car1,car2,car3 : char);

{cfr annexe 7.6 }


var i : integer;

begin
  if car2 <> 'd'
     then
        begin                                {parenthese}
          out_put.caract[out] := chr(27); {<esc>9}
          out_put.caract[out + 1] := '9';
          out := out + 2;
          premier(out_put,in_put,identif,out,inn,car1);
          pmorceau(out_put,in_put,identif,out,inn,'d');
        end
     else premier(out_put,in_put,identif,out,inn,car1); {accolade ou crochet}
  for i := 2 to ( compt - 1) do pmorceau(out_put,in_put,identif,out,inn,'d');
  pmorceau(out_put,in_put,identif,out,inn,car2);{milieu de la parenthese ou }
  for i := (compt + 1) to (2 * compt -1) do
                  pmorceau(out_put,in_put,identif,out,inn,'d');
  dernier(out_put,in_put,identif,out,inn,car3);
end;


{*************************************************************************}

                {PROCEDURE GDEPARENTH}

{*************************************************************************}


procedure gdeparenth(var out_put,in_put : chainevar;
                     var identif : chainentier; var out,inn : integer);

{cfr annexe 7.5 }

{La sequence a modifier est du type
<ESC>9...<ESC>9 !^H<ESC>8<ESC>8...!^H<ESC>8<ESC>8 <ESC>9...<ESC>9
      n-1                              n                      n+1
}
```

```
var ttype  : char;
    compt,i: integer;

begin
  ttype := in_put.caract[inn + 1];
  case ttype of
    'a':{bold on}
        cde3(out_put,in_put,identif,out,inn,4,'b','d');
    'b':{bold off}
        cde3(out_put,in_put,identif,out,inn,4,'b','@');
    'c':{draft on}
        begin
        cde3(out_put,in_put,identif,out,inn,4,'a','A');
        policecour := 2;
        end;
    'd':{italic on}
        begin
        cde3(out_put,in_put,identif,out,inn,4,'a','D');
        policecour := 5;
        end;
    'e':{regular on}
        begin
        cde3(out_put,in_put,identif,out,inn,4,'a','G');
        policecour := 8;
        end;
    'f':{draft,italic,regular off}
        begin
        cde3(out_put,in_put,identif,out,inn,4,'a','@');
        policecour := 1;
        end;
    '(',')','[',']','{','}','|' :
        begin
        for i:= 1 to 4 do  ignore(identif,inn,-1);
        compt := 1;
        while in_put.caract[inn] = chr(27) do   {<ESC>9}
          begin
                copy_maj(out_put,in_put,identif,out,inn,-1,2);
                compt := compt + 1;
          end;
        {dans compt on a le nombre de |^H<ESC>8<ESC>8}
        for i := 1 to (6 * compt) do ignore(identif,inn,-1);
        if compt = 1
          then
            begin
              out_put.caract[out] := ttype;
              out := out + 1;
              for i := 1 to 4 do ignore(identif,inn,-1); {<ESC>9<ESC>9}
            end
          else
              case ttype of
                '(': accolade(out_put,in_put,identif,out,inn,compt,'B','d','A');
                ')': accolade(out_put,in_put,identif,out,inn,compt,'c','d','b');
                '[': accolade(out_put,in_put,identif,out,inn,compt,'E','d','D');
                ']': accolade(out_put,in_put,identif,out,inn,compt,'F','d','e');
                '{': accolade(out_put,in_put,identif,out,inn,compt,'B','/','A');
                '}': accolade(out_put,in_put,identif,out,inn,compt,'c','C','b');
                '|': accolade(out_put,in_put,identif,out,inn,compt,' ','d',' ');
                end;
        end;
  end;
end;


{*************************************************************************}

                {PROCEDURE TRT_SANDERS}

{*************************************************************************}

procedure trtsanders(var in_put,out_put : chainevar ;

                     var policecour : integer);
{cfr annexe 7.4 }

var inn,out : integer; {position dans le buffer d'entree,de sortie}
    index : integer;  {no de l'action selon le caractere traite}
    identif : chainentier; {utilise pour traitr les <BS>}

begin
  inn :=1;
  out := 1;
  while inn <= in_put.long do
```

```
        begin {selection de l'action selon le caractere considere de l'input}
          index := action[ord(in_put.caract[inn]) + 1];
          if (index = 2) and ( in_put.caract[inn + 2] <> chr(15))
             then {CTRL\N : ce n'est pas une commande de nroff(^N-ch-^O)
                   C'est un caractere venant du micro-bee (^N ou ^N^Nch^O)}
                  begin
                    index := 1; {trt caractere normal}
                    if in_put.caract[inn + 1] = chr(14)
                       then ignore(identif,inn,-1); {si on a 2 CTRL\N conse-
                       cutifs pour differencier ce caractere de l'instructior
                       nroff,on en elimine 1 en output}
                  end;
          if (index = 9) and (in_put.caract[inn + 1] <> '|') then index :=1;
          if (index = 10) and ((in_put.caract[inn + 2] <> chr(8))
                or (in_put.caract[inn + 3] <> chr(14)))
                then index := 1; {pas la seq ^O char ^H ^N}


          case index of
                1: copy_maj(out_put,in_put,identif,out,inn,policecour,1);
                2: pnroff(out_put,in_put,identif,out,inn);
                   {2eme jeu de caracteres de nroff - suite du type ^N-ch-^O}
                3: pescape(out_put,in_put,identif,out,inn);
                   {commande "<ESC>..."}
                4: pbackspace(out_put,in_put,identif,out,inn,0);
                   {suite de backspaces a transformer}
                5: gdeparenth(out_put,in_put,identif,out,inn);
                   {modification des parentheses}
                6: copy_maj(out_put,in_put,identif,out,inn,-1,1);
                   {caractere recopie a ignorer pour le traitement des <BS>}
                7: underscore(out_put,in_put,identif,out,inn,policecour);
                   {commandes de deplacement vertical a rajouter peut-etre}
                8: signespec(out_put,in_put,identif,out,inn);
                   {signe speciaux a transformer}
                9: racinecarre(out_put,in_put,identif,out,inn);
                   {modification de la racine carree}
        end;
  end;
  out_put.long := out - 1;
end;


{*************************************************************************}

                {PROCEDURE INITIALISATION}

{*************************************************************************}

{cfr annexe 7.4 }
procedure init1;


var i : integer;

begin
  {FONT
   ----}
  for i := 1 to ASCII_COUNT do
        begin
          font[i,1] := 84;
          font[i,2] := 100;
        end;
  font[1,3] := 60;
  font[1,4] := 60;
  font[2,3] := 36;
  font[2,4] := 44;
  font[3,3] := 63;
  font[3,4] := 56;
  font[4,3] := 78;
  font[4,4] := 79;
  font[5,3] := 68;
  font[5,4] := 80;
  font[6,3] := 92;
  font[6,4] := 93;
  font[7,3] := 72;
  font[7,4] := 76;
  font[8,3] := 26;
  font[8,4] := 32;
  font[9,3] := 37;
  font[9,4] := 61;
  font[10,3] := 37;
  font[10,4] := 50;
```

```
font[11,3] := 62;
font[11,4] := 48;
font[12,3] := 62;
font[12,4] := 61;
font[13,3] := 45;
font[13,4] := 32;
font[14,3] := 62;
font[14,4] := 46;
font[15,3] := 36;
font[15,4] := 32;
font[16,3] := 52;
font[16,4] := 64;
font[17,3] := 62;
font[17,4] := 76;
font[18,3] := 72;
font[18,4] := 76;
font[19,3] := 72;
font[19,4] := 76;
font[20,3] := 72;
font[20,4] := 76;
font[21,3] := 72;
font[21,4] := 76;
font[22,3] := 72;
font[22,4] := 76;
font[23,3] := 72;
font[23,4] := 76;
font[24,3] := 72;
font[24,4] := 76;
font[25,3] := 72;
font[25,4] := 76;
font[26,3] := 72;
font[26,4] := 76;
font[27,3] := 36;
font[27,4] := 30;
font[28,3] := 44;
font[28,4] := 40;
font[29,3] := 68;
font[29,4] := 69;
font[30,3] := 64;
font[30,4] := 62;
font[31,3] := 74;
font[31,4] := 64;
font[32,3] := 62;
font[32,4] := 62;
font[33,3] := 60;
font[33,4] := 60;
font[34,3] := 76;
font[34,4] := 74;
font[35,3] := 78;
font[35,4] := 74;
font[36,3] := 79;
font[36,4] := 79;
font[37,3] := 82;
font[37,4] := 79;
font[38,3] := 74;
font[38,4] := 82;
font[39,3] := 74;
font[39,4] := 74;
font[40,3] := 84;
font[40,4] := 83;
font[41,3] := 76;
font[41,4] := 82;
font[42,3] := 27;
font[42,4] := 35;
font[43,3] := 54;
font[43,4] := 64;
font[44,3] := 79;
font[44,4] := 69;
font[45,3] := 65;
font[45,4] := 57;
font[46,3] := 91;
font[46,4] := 96;
font[47,3] := 79;
font[47,4] := 83;
font[48,3] := 79;
font[48,4] := 86;
font[49,3] := 74;
font[49,4] := 74;
font[50,3] := 81;
font[50,4] := 86;
font[51,3] := 74;
font[51,4] := 78;
```

```
font[52,3] := 72;
font[52,4] := 79;
font[53,3] := 64;
font[53,4] := 64;
font[54,3] := 76;
font[54,4] := 79;
font[55,3] := 76;
font[55,4] := 64;
font[56,3] := 96;
font[56,4] := 97;
font[57,3] := 66;
font[57,4] := 76;
font[58,3] := 66;
font[58,4] := 64;
font[59,3] := 78;
font[59,4] := 74;
font[60,3] := 60;
font[60,4] := 60;
font[61,3] := 60;
font[61,4] := 60;
font[62,3] := 60;
font[62,4] := 60;
font[63,3] := 51;
font[63,4] := 50;
font[64,3] := 60;
font[64,4] := 60;
font[65,3] := 29;
font[65,4] := 36;
font[66,3] := 32;
font[66,4] := 32;
font[67,3] := 66;
font[67,4] := 66;
font[68,3] := 62;
font[68,4] := 64;
font[69,3] := 62;
font[69,4] := 74;
font[70,3] := 32;
font[70,4] := 32;
font[71,3] := 43;
font[71,4] := 49;
font[72,3] := 64;
font[72,4] := 72;
font[73,3] := 64;
font[73,4] := 60;
font[74,3] := 21;
font[74,4] := 36;
font[75,3] := 38;
font[75,4] := 43;
font[76,3] := 60;
font[76,4] := 60;
font[77,3] := 27;
font[77,4] := 34;
font[78,3] := 96;
font[78,4] := 97;
font[79,3] := 62;
font[79,4] := 66;
font[80,3] := 32;
font[80,4] := 32;
font[81,3] := 68;
font[81,4] := 70;
font[82,3] := 66;
font[82,4] := 72;
font[83,3] := 48;
font[83,4] := 52;
font[84,3] := 58;
font[84,4] := 61;
font[85,3] := 44;
font[85,4] := 36;
font[86,3] := 32;
font[86,4] := 32;
font[87,3] := 60;
font[87,4] := 60;
font[88,3] := 78;
font[88,4] := 82;
font[89,3] := 67;
font[89,4] := 64;
font[90,3] := 62;
font[90,4] := 62;
font[91,3] := 64;
font[91,4] := 58;
for i := 92 to ASCII_COUNT do
      begin
```

```
            font[i,3] := 60;
            font[i,4] := 60;
          end;

end;

  {SANDERS
   -------}
procedure init2;

var i : integer;

begin
  for i := 1 to ASCII_COUNT do
   begin
     with sanders[i] do
          begin
            length := 7;
            car[1] := chr(27);
            car[2] := 'a';
            car[5] := chr(27);
            car[6] := 'a';
            car[7] := '@';
          end;
   end;
   with sanders[1] do
          begin
            car[3] := '@'; {police non connue}
            car[4] := '?'; {caractere inconnu}
          end;
   with sanders[2] do
          begin
            car[3] := 'C';
            car[4] := 'U';
          end;
   with sanders[3] do
          begin
            car[3] := '@';
            car[4] := '?';
          end;
   with sanders[4] do
          begin
            car[3] := 'C';
            car[4] := 'U';
            car[5] := chr(27);
            car[6] := 'U';
            car[7] := chr(27);
            car[8] := '1';
            car[9] := '~';
            car[10] := '\';
            car[11] := chr(27);
            car[12] := 'D';
            car[13] := chr(27);
            car[14] := 'a';
            car[15] := '@';
            car[16] := '-';
            length  := 16;
          end;
   with sanders[19] do
          begin
            car[3] := 'C';
            car[4] := 'U';
            car[5] := chr(27);
            car[6] := 'U';
            car[7] := chr(27);
            car[8] := '1';
            car[9] := '~';
            car[10] := '\';
            car[11] := chr(27);
            car[12] := 'D';
            car[13] := chr(27);
            car[14] := 'a';
            car[15] := '@';
            car[16] := '-';
            length  := 16;
          end;
   with sanders[6] do
          begin
            car[3] := 'C';
            car[4] := 'U';
            car[5] := chr(27);
```

```
        car[6] := 'U';
        car[7] := chr(27);
        car[8] := '1';
        car[9] := '~';
        car[10] := '\';
        car[11] := chr(27);
        car[12] := 'D';
        car[13] := chr(27);
        car[14] := 'a';
        car[15] := '@';
        car[16] := '/';
        length  := 16;
      end;
  with sanders[7] do
      begin
        car[3] := 'C';
        car[4] := 'u';
        car[5] := chr(27);
        car[6] := 'U';
        car[7] := chr(27);
        car[8] := '1';
        car[9] := '~';
        car[10] := '\';
        car[11] := chr(27);
        car[12] := 'D';
        car[13] := chr(27);
        car[14] := 'a';
        car[15] := '@';
        car[16] := '/';
        length  := 16;
      end;
  with sanders[8] do
      begin
        car[3] := '@'; {police non connue}
        car[4] := '?'; {caractere inconnu}
      end;
  with sanders[9] do
      begin
        car[3] := 'C';
        car[4] := 'u';
        car[5] := chr(27);
        car[6] := 'U';
        car[7] := chr(27);
        car[8] := '1';
        car[9] := '~';
        car[10] := '\';
        car[11] := chr(27);
        car[12] := 'D';
        car[13] := chr(27);
        car[14] := 'a';
        car[15] := '@';
        car[16] := '-';
        car[17] := chr(27);
        car[18] := 'U';
        car[19] := chr(27);
        car[20] := '1';
        car[21] := '~';
        car[22] := '\';
        car[23] := chr(27);
        car[24] := 'D';
        car[25] := '/';
        length  := 25;
      end;
  with sanders[10] do
      begin
        car[3] := '@';
        car[4] := 'V';
        car[5] := chr(27);
        car[6] := 'U';
        car[7] := chr(27);
        car[8] := '1';
        car[9] := '~';
        car[10] := '1';
        car[11] := chr(27);
        car[12] := 'D';
        car[13] := '-';
        length  := 13;
      end;
  with sanders[11] do
      begin
        car[3] := '@';
        car[4] := ']';
```

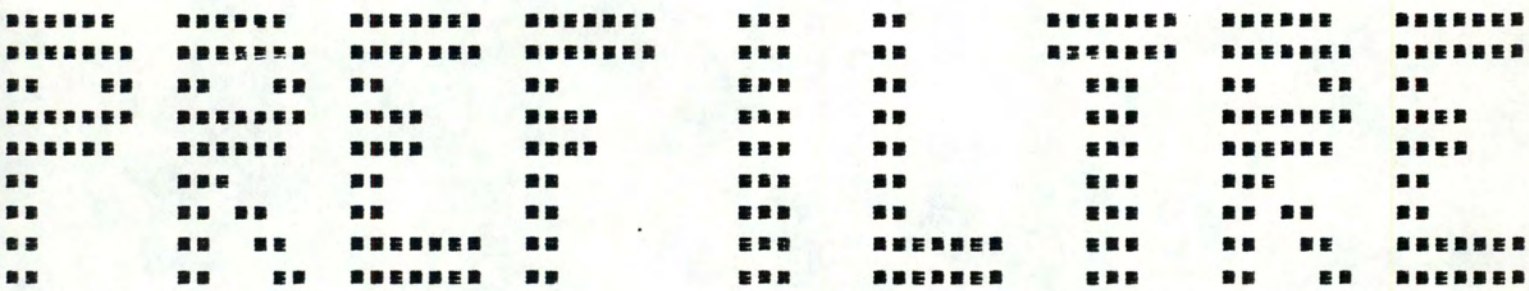
```
            car[5] := chr(27);
            car[6] := 'U';
            car[7] := chr(27);
            car[8] := '1';
            car[9] := '~';
            car[10] := '\';
            car[11] := chr(27);
            car[12] := 'D';
            car[13] := chr(27);
            car[14] := 'a';
            car[15] := 'C';
            car[16] := '=';
            car[17] := chr(27);
            car[18] := 'a';
            car[19] := '@';
            length  := 19;
          end;
    with sanders[12] do
          begin
            car[3] := '@';
            car[4] := ']';
            car[5] := chr(27);
            car[6] := 'U';
            car[7] := chr(27);
            car[8] := '1';
            car[9] := '~';
            car[10] := '\';
            car[11] := chr(27);
            car[12] := 'D';
            car[13] := chr(27);
            car[14] := 'a';
            car[15] := 'C';
            car[16] := '=';
            car[17] := chr(27);
            car[18] := 'a';
            car[19] := '@';
            car[20] := chr(27);
            car[21] := 'U';
            car[22] := chr(27);
            car[23] := '1';
            car[24] := '~';
            car[25] := '\';
            car[26] := chr(27);
            car[27] := 'D';
            car[28] := '/';
            length  := 28;
          end;
end;

procedure init3;

var i : integer;

begin
    with sanders[13] do
          begin
            car[3] := 'B';
            car[4] := '^';
          end;
    with sanders[14] do
          begin
            car[3] := 'C';
            car[4] := 'M';
          end;
    with sanders[15] do
          begin
            car[3] := 'C';
            car[4] := 'm';
          end;
    with sanders[16] do
          begin
            car[3] := 'C';
            car[4] := 'y';
          end;
    with sanders[17] do
          begin
            car[3] := 'C';
            car[4] := 't';
          end;
    with sanders[18] do
          begin
            car[3] := 'C';
```

```
        car[4] := 'u';
      end;
 with sanders[5] do
      begin
        car[3] := '@';
        car[4] := '?';
      end;
 with sanders[20] do
      begin
        car[3] := 'C';
        car[4] := 'U';
        car[5] := chr(27);
        car[6] := 'U';
        car[7] := chr(27);
        car[8] := '1';
        car[9] := '~';
        car[10] := '\';
        car[11] := chr(27);
        car[12] := 'D';
        car[13] := chr(27);
        car[14] := 'a';
        car[15] := '@';
        car[16] := '-';
        car[17] := chr(27);
        car[18] := 'U';
        car[19] := chr(27);
        car[20] := '1';
        car[21] := '~';
        car[22] := '\';
        car[23] := chr(27);
        car[24] := 'D';
        car[25] := '/';
        length  := 25;
      end;
for i := 21 to 32 do                {INCONNU : 'SPA' --> '?'}
 begin
   with sanders[i] do
      begin
        car[3] := '@'; {police non connue}
        car[4] := '?'; {caractere inconnu}
      end;
 end;
for i := 33 to 59 do
    with sanders[i] do car[3] := 'B'; {grec: '@'  --> 'Z'}
with sanders[60] do car[3] := 'C'; {'['}
with sanders[61] do car[3] := 'B'; {'\'}
with sanders[62] do car[3] := 'C'; {']'}
with sanders[63] do car[3] := 'B'; {^}
for i := 64 to 79 do                  {INCONNU DE '^' --> 'n'}
   with sanders[i] do
      begin
        car[3] := '@';
        car[4] := '?';
      end;
with sanders[80] do car[3] := 'C'; {'o'}
for i := 81 to 95 do                  {INCONNU DE 'p' --> '~'}
   with sanders[i] do
      begin
        car[3] := '@';
        car[4] := '?';
      end;
with sanders[33] do car[4] := 'n';
with sanders[34] do car[4] := 'a';
with sanders[35] do car[4] := 'b';
with sanders[36] do car[4] := 'z';
with sanders[37] do car[4] := 'd';
with sanders[38] do car[4] := 'L';
with sanders[39] do car[4] := 'V';
with sanders[40] do car[4] := 'C';
with sanders[41] do car[4] := 'Y';
with sanders[42] do car[4] := 't';
with sanders[43] do car[4] := 'q';
with sanders[44] do car[4] := 'r';
with sanders[45] do car[4] := 'l';
with sanders[46] do car[4] := 'm';
with sanders[47] do car[4] := 'h';
with sanders[48] do car[4] := 'I';
with sanders[49] do car[4] := 'G';
with sanders[50] do car[4] := 'g';
with sanders[51] do car[4] := 'F';
with sanders[52] do car[4] := 'e';
with sanders[53] do car[4] := 'i';
```

```
   with sanders[54] do car[4] := 'v';
   with sanders[55] do car[4] := 'u';
   with sanders[56] do car[4] := 'D';
   with sanders[57] do car[4] := 'o';
   with sanders[58] do car[4] := 's';
   with sanders[59] do car[4] := 'Z';
   with sanders[60] do car[4] := 'V';
   with sanders[61] do car[4] := 'c';
   with sanders[62] do car[4] := 'X';
   with sanders[63] do
         begin
          car[4] := chr(27);
          car[5] := 'U';
          car[6] := '*';
          car[7] := chr(27);
          car[8] := 'U';
          car[9] := chr(27);
          car[10] := '1';
          car[11] := '~';
          car[12] := '\';
          car[13] := chr(27);
          car[14] := 'D';
          car[15] := chr(27);
          car[16] := 'D';
          car[17] := '&';
          car[18] := chr(27);
          car[19] := 'U';
          car[20] := chr(27);
          car[21] := '1';
          car[22] := '~';
          car[23] := '\';
          car[24] := chr(27);
          car[25] := 'D';
          car[26] := chr(27);
          car[27] := 'D';
          car[28] := '!';
          car[29] := chr(27);
          car[30] := 'U';
          car[31] := chr(27);
          car[32] := 'a';
          car[33] := '@';
          length := 33;
        end;
   with sanders[80] do car[4] := 'z';

end;

   {ACTION
    ------}
procedure init4;

var i : integer;

begin
   action[1] := 6;
   action[2] := 6;
   action[3] := 1;
   action[4] := 6;
   action[5] := 6;
   action[6] := 1;
   action[7] := 1;
   action[8] := 6;
   action[9] := 4;
   action[10] := 6;
   action[11] := 6;
   action[12] := 1;
   action[13] := 6;
   action[14] := 6;
   action[15] := 2;
   action[16] := 5;
   action[17] := 1;
   action[18] := 6;
   action[19] := 1;
   action[20] := 6;
   action[21] := 1;
   action[22] := 1;
   action[23] := 1;
   action[24] := 6;
   action[25] := 1;
   action[26] := 1;
   action[27] := 1;
```

```
    action[28] := 3;
    action[29] := 6;
    action[30] := 1;
    action[31] := 1;
    action[32] := 6;
    for i := 33 to 43 do action[i] := 1;
    action[44] := 8; {+}
    for i := 45 to 47 do action[i] := 1;
    action[48] := 8; {/}
    for i := 49 to 60 do action[i] := 1;
    action[61] := 8; {<}
    action[62] := 8; {=}
    action[63] := 8; {>}
    for i := 64 to 92 do action[i] := 1;
    action[93] := 9; {\}
    for i := 94 to 95 do action[i] := 1;
    action[96] := 7;
    for i := 97 to 126 do action[i] := 1;
    action[127] := 8; {~}
    action[128] := 6;


    {ESCAPE
    -------}

    for i := 1 to 97 do escape[i] := 0;
    escape[11] := 100; {LF}
    escape[27] := 101; {SUB}
    escape[51] := 100; {2}
    escape[56] := 100; {7}
    escape[57] := 100; {8}
    escape[58] := 100; {9}
    escape[67] := 103; {B}
    escape[68] := 103; {C}
    escape[69] := 100; {D}
    escape[74] := 103; {I}
    escape[76] := 101; {K}
    escape[79] := 300; {N}
    escape[80] := 101; {O}
    escape[81] := 100; {P}
    escape[85] := 103; {T}
    escape[86] := 100; {U}
    escape[87] := 101; {V}
    escape[98] := 201; {a}
    escape[99] := 101; {b}
    escape[100] := 0;
    escape[101] := 109; {d}
    escape[102] := 103; {e}
    escape[103] := 103; {f}
    escape[104] := 103; {g}
    escape[105] := 102; {h}
    escape[106] := 103; {i}
    escape[107] := 105; {j}
    escape[108] := 102; {k}
    escape[109] := 102; {l}
    escape[110] := 0;
    escape[111] := 101; {n}
    escape[112] := 102; {o}
    escape[113] := 110; {p}
    escape[114] := 110; {q}
    escape[115] := 101; {r}
    escape[116] := 102; {s}
    escape[117] := 100; {t}
    escape[118] := 101; {u}
    escape[119] := 102; {v}
    escape[120] := 108; {w}
    escape[121] := 101; {x}
    escape[122] := 100; {y}
    escape[123] := 0;
    escape[124] := 0;
    escape[125] := 0;
    escape[126] := 0;
    escape[127] := 0;
    escape[128] := 0;

{POLICECOUR}

  policecour := 1; {1er jeu selectionne}

end;
```

```
{*************************************************************************}
                    {PROGRAMME PRINCIPAL}
{*************************************************************************}
{cfr annexe 7.3 }
begin
   init1;
   init2;
   init3;
   init4;
   reset(textin);
   rewrite(textout);
   while not eof(textin) do
         begin
              i := 1;
              test := 0;
              pascde := true;
              while (pascde = true ) and (not eoln(textin)) do
                 begin
                   if policecour <> 1 then pascde := false;
                   read(textin,ch);
                   in_put.caract[i] := ch;
                   index := action[ord(in_put.caract[i]) + 1];
                   case index of
                    1,7,8,9:;
                 2,3,5,6,10: pascde := false;
                          4: if test = i-1
                                then pascde := false {2 <BS> consecutifs}
                                else test := i;
                   end;
                   i := i + 1;
                 end;
              while not eoln(textin) do
                 begin
                   read(textin,ch);
                   in_put.caract[i] := ch;
                   i := i + 1;
                 end;
              in_put.long := i - 1;

              if pascde = true
                 then for i := 1 to in_put.long do
                         write(textout,in_put.caract[i])
                 else
                   begin
                     trtsanders(in_put,out_put,policecour);
                     for i := 1 to out_put.long do
                         write(textout,out_put.caract[i]);
                   end;

              readln(textin);
              writeln(textout);
         end;
end.
```



prefiltre.p

```
program prefiltre(textin,textout);

{cfr chapitre 4.11}
const MAXWORDLENGTH = 20;
      TABLESIZE = 25;
      BUFSIZE = 200;

type chainecar = packed array[1..BUFSIZE] of char;
     chainevar = record
                   caract : chainecar;
                   long   : integer;
                   end;
     wordtype = packed array[1..MAXWORDLENGTH] of char;
     entrytype = record
                   caract : wordtype;
                   action : integer;
                   end;
     tabletype = array[1..TABLESIZE] of entrytype;

var textin,textout : text;
    in_put : chainevar;
    ch,d1,d2 : char;
    delimd1d2,delimEGEN,bold_bool,font_bool : boolean;
    tampon : entrytype; {pour un mot}
    macrotable,table : tabletype;
    delimiteur : set of char;
    nbre_bool,nbre_font,longtable,longmacrotable : integer;

{*****************************************************************}

                {PROCEDURE PSEARCHWORD}

{*****************************************************************}

procedure psearchword(var table : tabletype; var tampon : entrytype;
                      var wordfound : boolean; var e,longtable : integer);

{cfr annexe 7.40 }
begin
  e := 1;
  wordfound := false;
  while (not wordfound) and (e <= TABLESIZE) do
        if table[e].caract = tampon.caract
           then wordfound := true
           else e := e + 1;
end;

{*****************************************************************}

                {PROCEDURE CP_BUF_OUT}

{*****************************************************************}

procedure cp_buf_out(var tampon : entrytype; var textout : text);

{cfr annexe 7.39 }
var j : integer;

begin
  for j :=1 to tampon.action do
      write(textout,tampon.caract[j]);
end;


{*****************************************************************}

                {PROCEDURE CP_INPUT_BUF}

{*****************************************************************}

procedure cp_input_buf(var in_put : chainevar; var tampon : entrytype;
                       var i : integer; d2 : char);

{cfr annexe 7.39 }
var j,l : integer;

begin
  j := 1;
  while (in_put.caract[i] in delimiteur) and (i <= in_put.long) do
    begin
        write(textout,in_put.caract[i]);
        i := i + 1;
```

```
    end;
  while (not (in_put.caract[i] in delimiteur)) and (in_put.caract[i] <> d2)
  and (i <= in_put.long) do
    begin
        tampon.caract[j] := in_put.caract[i];
        i := i + 1;
        j := j + 1;
    end;
  tampon.action := j - 1;
  for l := j to MAXWORDLENGTH do tampon.caract[l] := ' ';
end;

{************************************************************************}

                {PROCEDURE PREBOLD}

{************************************************************************}


procedure prebold(var textout : text;var in_put : chainevar;
                  var tampon : entrytype; var i,nbre_bool:integer;
                  car1 : char; var bold_bool : boolean);

{cfr annexe 7.38 }
var j : integer;

begin
  bold_bool := true;
  write(textout,chr(15),car1,chr(8),chr(14));
  cp_input_buf(in_put,tampon,i,' ');
  if tampon.action = 0
        then nbre_bool := 1
        else
         begin
          nbre_bool := ord(tampon.caract[1]) - 48;
          for j := 2 to tampon.action do
            nbre_bool := nbre_bool * 10 + ord( tampon.caract[j]) - 48;
         end;
end;

{************************************************************************}

        {PROCEDURE PACTIONMACRO}

{************************************************************************}

procedure pactionmacro(var macrotable : tabletype; var tampon : entrytype;
                       var textout : text; var delimEQEN : boolean;
                       var e,i,nbre_bool,nbre_font : integer;
                       var bold_bool,font_bool : boolean);

{cfr annexe 7.37 }
begin
  case macrotable[e].action of
    1 : {.EQ}
        begin
          delimEQEN := true;
          cp_buf_out(tampon,textout);
        end;
    2 : {.EN}
        begin
          delimEQEN := false;
          cp_buf_out(tampon,textout);
        end;
    3 : {.BD N}
        prebold(textout,in_put,tampon,i,nbre_bool,'a',bold_bool);
    4 : {.D N}
        prebold(textout,in_put,tampon,i,nbre_font,'c',font_bool);
    5 : {.I N}
        prebold(textout,in_put,tampon,i,nbre_font,'d',font_bool);
    6 : {.HR N}
        prebold(textout,in_put,tampon,i,nbre_font,'e',font_bool);
  end;
end;

{************************************************************************}

                {PROCEDURE CP_LI_INPUT}

{************************************************************************}

procedure cp_li_input(var textin : text;var in_put : chainevar;var ch : char;
```

```
                        valinit : integer);

{cfr annexe 7.36 }
var j : integer;

begin
  j := valinit;
  while not eoln(textin) do
     begin
        read(textin,ch);
        in_put.caract[j] := ch;
        j := j + 1;
     end;
  in_put.long := j - 1;
end;

{*****************************************************************}

                {PROCEDURE TRTMACRO}

{*****************************************************************}

procedure trtmacro(var macrotable : tabletype; var tampon : entrytype;
                   var textout,textin : text; var delimEGEN : boolean;
                   var in_put : chainevar;var nbre_bool,nbre_font : integer;
                   var longtable :integer;var bold_bool,font_bool : boolean);

{cfr annexe 7.35 }
var wordfound : boolean;
    e,i : integer;

begin
  i:=1;
  while i <= in_put.long do
    begin
      cp_input_buf(in_put,tampon,i,' ');
      if tampon.action > 0
        then
         begin
          psearchword(macrotable,tampon,wordfound,e,longmacrotable);
          if wordfound = true
             then pactionmacro(macrotable,tampon,textout,delimEGEN,e,i,
                  nbre_bool,nbre_font,bold_bool,font_bool)
             else cp_buf_out(tampon,textout);
         end;
    end;
  readln(textin);
  writeln(textout);
end;

{*****************************************************************}

                {PROCEDURE CP_LI_OUT}

{*****************************************************************}

procedure cp_li_out(var textin,textout : text; var ch : char);

{cfr annexe 7.35 }
begin
  if not eoln(textin) then write(textout,ch);
  while not eoln(textin) do
    begin
        read(textin,ch);
        write(textout,ch);
    end;
  readln(textin);
  writeln(textout);
end;


{*****************************************************************}

                {PROCEDURE PLEFTRIGHT}

{*****************************************************************}

procedure pleftright(var in_put : chainevar; var tampon : entrytype;
                   var textout : text; var i : integer);

{cfr annexe 7.34 }
```

```
var l : integer;

begin
  l := tampon.action + 2;
  tampon.caract[l - 1] := ' ';
  while in_put.caract[i] = ' ' do i := i + 1;
  tampon.caract[l] := in_put.caract[i] ;
  i := i + 1;
  tampon.action := l;
  case tampon.caract[l] of
'{','}'   : write(textout,chr(15),'"',tampon.caract[l],'"',chr(8),chr(14),' ');
'|','[',']','(',')': write(textout,chr(15),tampon.caract[l],chr(8),chr(14),' '
  end;
  cp_buf_out(tampon,textout);
end;

{************************************************************************}

                {PROCEDURE PDELIM}

{************************************************************************}

procedure pdelim(var in_put :chainevar; var tampon : entrytype;
                 var textout : text; var i : integer; var d1,d2 : char;
                 var delimd1d2 : boolean);

{cfr annexe 7.33 }
begin
  cp_buf_out(tampon,textout);
  cp_input_buf(in_put,tampon,i,d2);
  if tampon.caract = 'off                    '
     then
        begin
          delimd1d2 := false;
          d2 := ' ';
        end
     else
        begin
          delimd1d2 := true;
          d1 := tampon.caract[1];
          d2 := tampon.caract[2];
        end;
  cp_buf_out(tampon,textout);
end;

{************************************************************************}

                {PROCEDURE PACTIONCLE}

{************************************************************************}

procedure pactioncle(var table : tabletype; var in_put : chainevar;
                     var tampon : entrytype; var textout : text;
                     var k,i : integer; var d1,d2 : char;
                     var delimd1d2 : boolean);

{cfr annexe 7.32 }
begin
  case table[k].action of
     1 : {left ou right}
         pleftright(in_put,tampon,textout,i);
     2 : {delim}
         pdelim(in_put,tampon,textout,i,d1,d2,delimd1d2);
     3 : write(textout,chr(14),'!',chr(15));
     4 : write(textout,chr(14),'1',chr(15));
     5 : write(textout,chr(14),'#',chr(15));
     6 : write(textout,chr(14),'2',chr(15));
     7 : write(textout,chr(14),'%',chr(15));
     8 : write(textout,chr(14),'&',chr(15));
     9 : write(textout,chr(14),'3',chr(15));
    10 : write(textout,chr(14),'(',chr(15));
    11 : write(textout,chr(14),')',chr(15));
    12 : write(textout,chr(14),'*',chr(15));
    13 : write(textout,chr(14),'+',chr(15));
    14 : write(textout,chr(14),',',chr(15));
    15 : write(textout,chr(14),'-',chr(15));
    16 : write(textout,chr(14),'.',chr(15));
    17 : write(textout,chr(14),'/',chr(15));
    18 : write(textout,chr(14),'0',chr(15));
  end;
end;
```

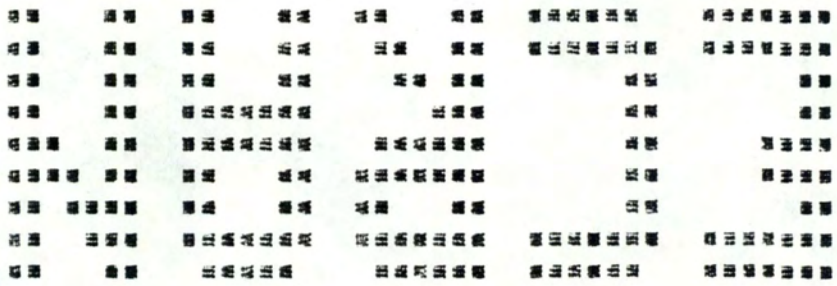
```
{*****************************************************************}

                {PROCEDURE PBOLD}

{*****************************************************************}

procedure pbold(var textout : text; var bold_bool : boolean;
        var nbre_bool : integer;car2 : char);

{cfr annexe 7.31 }
begin
  if bold_bool then
     begin
        if nbre_bool <> 1
          then nbre_bool := nbre_bool - 1
          else begin
                bold_bool := false;
                writeln(textout,chr(15),car2,chr(8),chr(14));
               end;
     end;
end;

{*****************************************************************}

                {PROCEDURE TRTLIGNEMATH}

{*****************************************************************}

procedure trtlignemath(var textout,textin : text; var in_put : chainevar;
                       var tampon : entrytype; var table : tabletype;
                       var d1,d2 : char; var delimd1d2 : boolean;
                       var nbre_bool,nbre_font,longtable : integer;
                       var bold_bool,font_bool : boolean);

{cfr annexe 7.29 }
var wordfound : boolean;
    i,k : integer;

begin
  i := 1;
  while i <= in_put.long  do
   begin
     cp_input_buf(in_put,tampon,i,' ');
     if tampon.action > 0
        then
         begin
           psearchword(table,tampon,wordfound,k,longtable);
           if wordfound = true
              then pactioncle(table,in_put,tampon,textout,k,i,d1,d2,delimd1d2)
              else cp_buf_out(tampon,textout);
         end;
   end;
  readln(textin);
  writeln(textout);
  pbold(textout,bold_bool,nbre_bool,'b');
  pbold(textout,font_bool,nbre_font,'f');
end;

{*****************************************************************}

                {PROCEDURE TRTDELIM}

{*****************************************************************}

procedure trtdelim(var textout,textin : text; var in_put : chainevar;
                   var tampon : entrytype; var table : tabletype;
                   var d1,d2 : char; var delimd1d2 : boolean;
                   var nbre_bool,nbre_font,longtable : integer;
                   var bold_bool,font_bool : boolean);

{cfr annexe 7.30 }
var wordfound : boolean;
    i,k : integer;

begin
  i := 1;

  while i <= in_put.long do
   begin
    while (in_put.caract[i] <> d1) and (i <= in_put.long) do
        begin
          write(textout,in_put.caract[i]);
```

```
          i := i + 1;
        end;
    if (in_put.caract[i] = d1) and (i <= in_put.long) then
        begin
          write(textout, in_put.caract[i]);
          i := i + 1;
        end;
    while (in_put.caract[i] <> d2) and (i <= in_put.long) do
     begin
      cp_input_buf(in_put, tampon, i, d2);
      if tampon.action > 0
        then
          begin
          psearchword(table, tampon, wordfound, k, longtable);
          if wordfound = true
             then pactioncle(table, in_put, tampon, textout, k, i, d1, d2, delimd1d2)
             else cp_buf_out(tampon, textout);
          end;


     end;
    if (in_put.caract[i] = d2) and (i <= in_put.long) then
        begin
          write(textout, in_put.caract[i]);
          i := i + 1;
        end;
   end;
  readln(textin);
  writeln(textout);
  pbold(textout, bold_bool, nbre_bool, 'b');
  pbold(textout, font_bool, nbre_font, 'f');
end;

{************************************************************************}

                {PROCEDURE PCAS1}

{************************************************************************}

procedure pcas1(var textin, textout : text; var ch : char;
                var delimd1d2, delimEGEN : boolean;
                var in_put : chainevar; var tampon : entrytype;
                var nbre_bool, nbre_font : integer;
                var bold_bool, font_bool : boolean);

{cfr annexe 7.26 }
begin
   if not eoln(textin) then read(textin, ch);
   if ch = '.'
      then
        begin
          in_put.caract[1] := ch;
          cp_li_input(textin, in_put, ch, 2);
          trtmacro(macrotable, tampon, textout, textin, delimEGEN, in_put,
                   nbre_bool, nbre_font, longmacrotable, bold_bool, font_bool);
        end
      else
        begin
         cp_li_out(textin, textout, ch);
         pbold(textout, bold_bool, nbre_bool, 'b');
         pbold(textout, font_bool, nbre_font, 'f');
        end;
end;

{************************************************************************}

                {PROCEDURE PCAS2}

{************************************************************************}

procedure pcas2(var textin, textout : text; var ch, d1, d2 : char;
                var delimd1d2, delimEGEN : boolean;
                var in_put : chainevar;  var tampon : entrytype;
                var nbre_bool, nbre_font : integer;
                var bold_bool, font_bool : boolean);

{cfr annexe 7.27 }
begin
   cp_li_input(textin, in_put, ch, 1);
   if in_put.caract[1] = '.'
      then trtmacro(macrotable, tampon, textout, textin, delimEGEN, in_put,
                   nbre_bool, nbre_font, longmacrotable, bold_bool, font_bool)
```

```
      else trtlignemath(textout,textin,in_put,tampon,table,d1,d2,delimd1d2,
                        nbre_bool,nbre_font,longtable,bold_bool,font_bool);
end;

{*************************************************************************}

                    {PROCEDURE PCAS3}

{*************************************************************************}

procedure pcas3(var textin,textout : text; var ch,d1,d2 : char;
                var delimd1d2,delimEQEN : boolean;
                var in_put : chainevar;  var tampon : entrytype;
                var nbre_bool,nbre_font : integer;
                var bold_bool,font_bool : boolean);

{cfr annexe 7.28 }
begin
   cp_li_input(textin,in_put,ch,1);
   if in_put.caract[1] = '.'
      then trtmacro(macrotable,tampon,textout,textin,delimEQEN,in_put,
                        nbre_bool,nbre_font,longmacrotable,bold_bool,font_bool)
      else trtdelim(textout,textin,in_put,tampon,table,d1,d2,delimd1d2,
                        nbre_bool,nbre_font,longtable,bold_bool,font_bool);
end;

{*************************************************************************}

                    {PROCEDURE INITIALISATION}

{*************************************************************************}

procedure initialisation;
{cfr annexe 7.26 }
{        table                                    macrotable
    caract  |  action                          caract   | action
   ---------------------                     ------------------
    left    |    1                           .EQ       |   1
    right   |    1                           .EN       |   2
    delim   |    2                           .BO       |   3
    inc     |    3                           .D        |   4
    comp    |    4                           .I        |   5
    app     |    5                           .HR       |   6
    cont    |    6
    ninc    |    7
    ncomp   |    8
    napp    |    9
    ncont   |   10
   pourtout |   11
    ilexist |   12
   nilexist |   13
    /\      |   14
    <|      |   15
    >|      |   16
    dbar    |   17
    inter   |   18
}

begin
  longtable := 19;
  longmacrotable := 6;
  delimd1d2 := false;
  delimEQEN := false;
  d1 := ' ';
  d2 := ' ';
  bold_bool := false;
  font_bool := false;
  delimiteur := [' ',',',';','''','-',':','!','?','{','}','[',']','~','"'];
  with table[1] do
    begin
        caract := 'left                ';
        action := 1;
    end;
  with table[2] do
    begin
        caract := 'right               ';
        action := 1;
    end;
  with table[3] do
    begin
        caract := 'delim               ';
        action := 2;
```

```
  end;
with table[4] do
  begin
      caract := 'inc                    ';
      action := 3;
  end;
with table[5] do
  begin
      caract := 'comp                   ';
      action := 4;
  end;
with table[6] do
  begin
      caract := 'app                    ';
      action := 5;
  end;
with table[7] do
  begin
      caract := 'cont                   ';
      action := 6;
  end;
with table[8] do
  begin
      caract := 'ninc                   ';
      action := 7;
  end;
with table[9] do
  begin
      caract := 'ncomp                  ';
      action := 8;
  end;
with table[10] do
  begin
      caract := 'napp                   ';
      action := 9;
  end;
with table[11] do
  begin
      caract := 'ncont                  ';
      action := 10;
  end;
with table[12] do
  begin
      caract := 'pourtout               ';
      action := 11;
  end;
with table[13] do
  begin
      caract := 'ilexist                ';
      action := 12;
  end;
with table[14] do
  begin
      caract := 'nilexist               ';
      action := 13;
  end;
with table[15] do
  begin
      caract := '/\                     ';
      action := 14;
  end;
with table[16] do
  begin
      caract := '<|                     ';
      action := 15;
  end;
with table[17] do
  begin
      caract := '>|                     ';
      action := 16;
  end;
with table[18] do
  begin
      caract := 'dbar                   ';
      action := 17;
  end;
with table[19] do
  begin
      caract := 'inter                  ';
      action := 18;
  end;
with macrotable[1] do
```

```
    begin
        caract := '.EQ                  ';
        action := 1;
    end;
  with macrotable[2] do
    begin
        caract := '.EN                  ';
        action := 2;
    end;
  with macrotable[3] do
    begin
        caract := '.BO                  ';
        action := 3;
    end;
  with macrotable[4] do
    begin
        caract := '.D                   ';
        action := 4;
    end;
  with macrotable[5] do
    begin
        caract := '.I                   ';
        action := 5;
    end;
  with macrotable[6] do
    begin
        caract := '.HR                  ';
        action := 6;
    end;
end;

{*************************************************************************}

                {PROGRAMME PRINCIPAL}

{*************************************************************************}

{cfr annexe 7.25 }
begin
  initialisation;
  reset(textin);
  rewrite(textout);
  while not eof(textin) do
    begin
      if delimd1d2
        then
          while (not delimEQEN) and (not eof(textin))
          do pcas3(textin,textout,ch,d1,d2,delimd1d2,delimEQEN,in_put,tampon,
                nbre_bool,nbre_font,bold_bool,font_bool)
        else
          while (not delimEQEN) and (not eof(textin))
          do pcas1(textin,textout,ch,delimd1d2,delimEQEN,in_put,tampon,
                nbre_bool,nbre_font,bold_bool,font_bool);
      while delimEQEN and (not eof(textin))
      do pcas2(textin,textout,ch,d1,d2,delimd1d2,delimEQEN,in_put,tampon,
                nbre_bool,nbre_font,bold_bool,font_bool);
    end;
end.
```



```
program ecran(textin,textout,output);

{cfr chapitre 4.13}

const BUFSIZE = 800; {longueur maximale du tampon}
      NBRE_ARG_DFLT = 2;
      ASCII_COUNT = 95; {nombre de caracteres ascii imprimables}
      FONT_COUNT = 4; {nombre de polices dont la largeur des caracteres
                       different}
      SEQ_SIZE = 40; {longueur maximale de la sequence sanders correspondant
                      a un caractere du 2eme jeu decaracteres de nroff}
      TOTAL_ASCII = 128; {total des caracteres ascii}



type chainecar = packed array[1..BUFSIZE] of char; {representation du tampon}
     chainevar = record
                 caract : chainecar ; {tampon}
                 long   : integer ;   {longueur effective du tampon}
                 end;
     lignecar = packed array[1.. SEQ_SIZE] of char;
     lignevar = record
                car    : lignecar ;
                length : integer ;
                end;
     sanroff = array[1..ASCII_COUNT] of lignevar;
                {contient la sequence sanders pour generer tout caractere
                 du 2eme jeu de nroff }
     selection = packed array[1..TOTAL_ASCII] of integer;
                 {- contient le numero du traitement a effectuer pour le
                    caractere donne.
                  - contient les specifications necessaires quand on
                    rencontre une commande escape.
                    chaque element est un entier de 3 chiffres :
                        - centaine = 1 si cde connue ;0 sinon
                        - dizaine  = 1 si la liste des arguments termine par 1
                                     point ; 0 sinon
                        - unite    = nombre d'arguments de la commande}
     chainentier = packed array[1..BUFSIZE] of integer; {utilise lorsque on
                   traite des backspaces.pour chaque caractere du tampon
                   d'entree,on a 1 valeur correspondante dans le tableau :
                        - 1 : no de police
                        -zero          : backspace
                        - -1 : commande a ignorer }


var textin,textout : text;
    in_put,out_put : chainevar;
    ch : char;
    i ,nbre_li ,nbre_li_blan : integer;
    monter,descendre,bool_uns : boolean;
    video : sanroff ; {sequence generation sur VT100}
    action : selection; {choix du traitement pour un caractere}
    escape : selection; {specification pour la commande <esc>...}

{*************************************************************************}

                {PROCEDURE ASCIIDEC}

{*************************************************************************}

procedure asciidec(var in_put : chainevar; var inn,nombre : integer;
                   nbrearg : integer);

{cfr annexe 7.61 }
var i : integer;

begin
  nombre := 0 ;
  for i := 1 to nbrearg do
      nombre := nombre * 64 + ord(in_put.caract[inn + i - 1]) -64;
end;

{*************************************************************************}

                {PROCEDURE IGNORE}

{*************************************************************************}
```

```
procedure ignore(var identif : chainentier; var inn : integer;
                 font_use : integer);

{cfr annexe 7.60 }

begin
  identif[inn] := font_use;
  inn := inn +1;
end;

{*************************************************************************}

                {PROCEDURE CDE2}

{*************************************************************************}

procedure cde2(var out_put,in_put : chainevar; var identif : chainentier;
               var out,inn : integer; nbreignore : integer; lettre : char);


{cfr annexe 7.59 }
var i : integer;

begin
  out_put.caract[out] := chr(27);
  out_put.caract[out + 1] := lettre;
  out := out + 2;
  for i := 1 to nbreignore do ignore(identif,inn,-1);
end;


{*************************************************************************}

                {PROCEDURE PMVT}

{*************************************************************************}

procedure pmvt(var out_put,in_put : chainevar; var identif : chainentier;
               var out,inn : integer; nbreignore,pn : integer; lettre : char);

{cfr annexe 7.57 }
var i : integer;

begin
  out_put.caract[out] := chr(27);
  out_put.caract[out + 1] := '[';
  out := out + 2;
  if pn > 9 then
     begin
        out_put.caract[out] := chr((pn div 10) + 48);
        out := out + 1;
     end;
  out_put.caract[out] := chr((pn mod 10) + 48);
  out_put.caract[out + 1] := lettre;
  out := out + 2;
  for i := 1 to nbreignore do ignore(identif,inn,-1);
end;

{*************************************************************************}

                {PROCEDURE PFONT}

{*************************************************************************}

procedure pfont(var out_put,in_put : chainevar; var identif : chainentier;
               var out,inn : integer; nbreignore : integer; lettre : char);

{cfr annexe 7.58 }
var i : integer;

begin
  out_put.caract[out] := chr(27);
  out_put.caract[out + 1] := '(';
  out_put.caract[out + 2] := lettre;
  out := out + 3;
  for i := 1 to nbreignore do ignore(identif,inn,-1);
end;

{*************************************************************************}

                {PROCEDURE COPY_MAJ}
```

```
{*************************************************************************}

procedure copy_maj(var out_put,in_put : chainevar;var identif : chainentier;
                   var out,inn : integer;incr : integer);

{cfr annexe 7.60 }
var k : integer;

begin
  for k := 1 to incr do
        begin
          out_put.caract[out + k -1] := in_put.caract[inn + k -1];
          identif[inn + k -1] := 1;
      end;
  inn := inn + incr;
  out := out + incr;
end;


{*************************************************************************}

                {PROCEDURE PNROFF}

{*************************************************************************}

procedure pnroff(var out_put,in_put : chainevar; var identif : chainentier;
                 var out,inn : integer);

{cfr annexe 7.57 }

var i,j : integer;

begin
  i := ord(in_put.caract[inn + 1]) - 31; {entree dans la table video}
  with video[i] do
    begin
      for j := 1 to length do out_put.caract[out + j -1] := car[j];
      out := out + length;
    end;
  ignore(identif,inn,-1);
  ignore(identif,inn,1);
  ignore(identif,inn,-1);
end;

{*************************************************************************}

                {PROCEDURE PESCAPE}

{*************************************************************************}

procedure pescape (var out_put,in_put : chainevar; var identif : chainentier ;
                   var out,inn : integer;
                   var monter,descendre,bool_uns : boolean);

{cfr annexe 7.55 }
var carcde,cde,dot,nbrearg,nombre,i:integer;

begin
  carcde := escape[ord(in_put.caract[inn + 1]) + 1];
  cde := carcde div 100; {quotient de la division par 100 : chiffre des cent.}
  case cde of
        0 : {commande inconnue : on ne recopie pas <esc> en output
                                 on l'ignore si <bs> a traiter.}
            ignore(identif,inn,-1);
        1 : {commande connue et ignoree}
            begin
              carcde := carcde mod 100; {reste : dizaine-unite}
              dot := carcde div 10; {chiffre des dizaines}
              if dot = 0 {cde dont la liste des arg. ne termine pas par "."}
                 then begin
                      nbrearg := carcde mod 10; {unite = #arg}
                      for i := 1 to (nbrearg + 2) do ignore(identif,inn,-1);
                      end
                 else begin
                      while in_put.caract[inn] <> '.' do
                        ignore(identif,inn,-1);
                        ignore(identif,inn,-1);
                      end;
            end;
        2 : begin
            case in_put.caract[inn + 1] of
               'N'          : pmvt(out_put,in_put,identif,out,inn,2,1,'D');
```

```
                '2'             : pmvt(out_put,in_put,identif,out,inn,2,3,'g');
                'U','9'         : begin
                                    descendre := true;
                                    cde2(out_put,in_put,identif,out,inn,2,'D');
                                  end;
                '7'             : begin
                                    monter := true;
                                    pmvt(out_put,in_put,identif,out,inn,2,2,'A');
                                  end;
                'D','8'         : begin
                                    monter := true;
                                    pmvt(out_put,in_put,identif,out,inn,2,1,'A');
                                  end;
             end;
            end;
        3 : {commande <ESC>u(n)}
            begin
              if bool_uns = false
                then begin
                     bool_uns := false;
                     pmvt(out_put,in_put,identif,out,inn,3,4,'m');
                     end
                else begin
                     bool_uns := true;
                     pmvt(out_put,in_put,identif,out,inn,3,0,'m');
                     end;
            end;
        4 : {<ESC>b(n)}
            if in_put.caract[inn + 2 ] = '@'
               then pmvt(out_put,in_put,identif,out,inn,3,0,'m')
               else pmvt(out_put,in_put,identif,out,inn,3,1,'m');
        5 : {<ESC>s(n)(a)}
            begin
               for i := 1 to 2 do ignore(identif,inn,-1);
               asciidec(in_put,inn,nombre,1);
               for i := 1 to nombre do
                  begin
                    out_put.caract[out + i - 1] := in_put.caract[inn + 1];
                    out := out + 1;
                  end;
               for i := 1 to 2 do ignore(identif,inn,-1);
            end;
        6 : {<ESC>h(nn)}
            begin
               for i := 1 to 2 do ignore(identif,inn,-1);
               asciidec(in_put,inn,nombre,2);
               pmvt(out_put,in_put,identif,out,inn,2,nombre,'C')
            end;
        7 : {<ESC>v(nn)}
            begin
               for i := 1 to 2 do ignore(identif,inn,-1);
               asciidec(in_put,inn,nombre,2);
               out_put.caract[out] := chr(27);
               out_put.caract[out + 1] := '[';
               out := out + 2;
               if nombre > 9 then
                       begin
                       out_put.caract[out] := chr((nombre div 10) + 48);
                       out := out + 1;
                       end;
               out_put.caract[out] := chr((nombre mod 10) + 48);
               out_put.caract[out + 1] := ';';
               out_put.caract[out + 2] := '1';
               out_put.caract[out + 3] := 'H';
               out := out + 4;
               for i := 1 to 2 do ignore(identif,inn,-1);
            end;
  end;
end;

{*************************************************************************}

                {PROCEDURE BARREFRACT}

{*************************************************************************}

procedure barrefract(var out_put,in_put : chainevar;var identif : chainentier;
                     var out,inn,bs_count,uns_count : integer);

{cfr annexe 7.54 }
var i : integer;
```

```
begin
  cde2(out_put,in_put,identif,out,inn,2,'D');
  pmvt(out_put,in_put,identif,out,inn,0,bs_count,'D');   {bs}
  pfont(out_put,in_put,identif,out,inn,0,'0');   {<ESC>(0}
  for i := 1 to uns_count do out_put.caract[out + i - 1] := 'q';
  out := out + uns_count;
  pfont(out_put,in_put,identif,out,inn,0,'A');   {<ESC>(A}
end;

{*****************************************************************}

                {PROCEDURE UNDERSCORE}

{*****************************************************************}

procedure underscore(var out_put,in_put : chainevar; var identif: chainentier;
                     var out,inn,bs_count,aux : integer);

{cfr annexe 7.53 }
var i,aux1,extra : integer;

begin
  aux1 := aux - 1;
  extra := bs_count;
  while bs_count <> 0 do
    begin
        aux := aux - 1;
        case identif[aux] of
           -1 :;
            0 :begin
                 bs_count := bs_count + 1;
                 extra := extra + 1;
               end;
            1 :bs_count := bs_count - 1;
        end;
    end;
    if aux > 2 then
      if (identif[aux - 1] = 0) and (identif[aux - 2] = 1)
       then aux := aux - 2;
    out := out - extra;
    pmvt(out_put,in_put,identif,out,inn,0,4,'m');  {<ESC>[4m}
    for i := aux to aux1 do out_put.caract[out + i - aux] := in_put.caract[i];
    out := out + aux1 - aux + 1;
    pmvt(out_put,in_put,identif,out,inn,0,0,'m');  {<ESC>[0m}
end;

{*****************************************************************}

                {PROCEDURE PBACKSPACE}

{*****************************************************************}

procedure pbackspace(var out_put,in_put : chainevar; var identif : chainentier
                     var out,inn : integer );

{cfr annexe 7.53 }
var bs_count,uns_count,aux,i: integer;

begin
  bs_count := 1;
  uns_count := 0;
  aux := inn;
  ignore(identif,inn,0);

  while (in_put.caract[inn] = chr(8)) and (inn <= in_put.long) do
        begin
          bs_count := bs_count + 1;
          ignore(identif,inn,0);
        end;
  while (in_put.caract[inn] = chr(27)) and (in_put.caract[inn + 1] = '9')
        and (inn <= in_put.long) do
        pescape(out_put,in_put,identif,out,inn,monter,descendre,bool_uns);
  while (in_put.caract[inn] = '_') and (inn <= in_put.long) do
        begin
          uns_count := uns_count + 1;
          ignore(identif,inn,1);
        end;
  if bs_count <= uns_count
     then
          if (in_put.caract[inn] = chr(27)) and (in_put.caract[inn + 1] = '9'
              and (inn <= in_put.long)
            then barrefract(out_put,in_put,identif,out,inn,bs_count,uns_count
```

```
            else underscore(out_put,in_put,identif,out,inn,bs_count,aux)
      else
         begin
           for i := aux  to (inn - 1) do
                 out_put.caract[out + i - aux] := in_put.caract[i];
           out := out + inn - aux;
         end;
end;

{*************************************************************************}

                  {PROCEDURE PGENERSIGN}

{*************************************************************************}

procedure pgenersign(var out_put,input : chainevar;var identif : chainentier;
                     var out,inn : integer; caract1,caract2 : char);

{cfr annexe 7.52 }
begin
  if (in_put.caract[inn + 1] = chr(8)) and (in_put.caract[inn + 2] = caract1)
     then
        begin
          ignore(identif,inn,-1);
          ignore(identif,inn,-1);
          ignore(identif,inn,1);
          pfont(out_put,in_put,identif,out,inn,0,'0');   {<ESC>(0}
          out_put.caract[out] := caract2;
          out := out + 1;
          pfont(out_put,in_put,identif,out,inn,0,'A');   {<ESC>(A}
        end
     else
        copy_maj(out_put,in_put,identif,out,inn,1);
end;



{*************************************************************************}

                  {PROCEDURE SIGNESPEC}

{*************************************************************************}

procedure signespec(var out_put,in_put : chainevar; var identif : chainentier;
                    var out,inn : integer);

{cfr annexe 7.51 }
begin
  case in_put.caract[inn] of
         '>' : pgenersign(out_put,in_put,identif,out,inn,'_','z');
         '<' : pgenersign(out_put,in_put,identif,out,inn,'=','y');
         '/' : pgenersign(out_put,in_put,identif,out,inn,'=','!');
         '+' : pgenersign(out_put,in_put,identif,out,inn,'_','g');
   end;
end;

{*************************************************************************}

                  {PROCEDURE RACINECARRE}

{*************************************************************************}

procedure racinecarre(var out_put,in_put : chainevar ;
                      var identif : chainentier ; var out,inn : integer);


{cfr annexe 7.50 }
{forme de la racine : \|^H<esc>8...|^H<esc>8|<esc>8<esc>8__.....__}

var bs_count,i : integer;

begin
  out_put.caract[out] := in_put.caract[inn]; {\}
  out := out + 1;
  pfont(out_put,in_put,identif,out,inn,0,'0');   {<ESC>(0}
  out_put.caract[out] := 'x';
  out := out + 1;
  ignore(identif,inn,1);{\}
  ignore(identif,inn,1);{|}
```

```
  while in_put.caract[inn] = chr(8) do          {^H<ESC>8!}
      begin
        out_put.caract[out] := in_put.caract[inn];
        ignore(identif,inn,0);
        out := out + 1;
        pmvt(out_put,in_put,identif,out,inn,2,1,'A');
        out_put.caract[out] := 'x';
        out := out + 1;
        ignore(identif,inn,1);
      end;

  if (in_put.caract[inn] = chr(27)) and (in_put.caract[inn +1] = '8')
    then    {<ESC>8<ESC>8}
      begin
        out_put.caract[out] := chr(8) ;{^H}
        out := out + 1;
        pmvt(out_put,in_put,identif,out,inn,2,1,'A');{<ESC>[1A = <ESC>8}
        out_put.caract[out] := '1';
        out := out + 1;
        for i := 1 to 2 do ignore(identif,inn,-1); {<ESC>8}
      end;


   while in_put.caract[inn] = '_' do
        begin
          ignore(identif,inn,1);
          out_put.caract[out] := 'q';
          out := out + 1;
        end;
   pfont(out_put,in_put,identif,out,inn,0,'A');   {<ESC>(A}
   bs_count := 0;
   while in_put.caract[inn] = chr(8) do
        begin
          bs_count := bs_count + 1;
          ignore(identif,inn,0);
        end;
   pmvt(out_put,in_put,identif,out,inn,0,bs_count,'D');

  if (in_put.caract[inn] = chr(27)) and (in_put.caract[inn +1] = '9')
       then for i := 1 to 2 do ignore(identif,inn,-1); {<ESC>9}
end;

{*************************************************************************}

                {PROCEDURE PREMIER}

{*************************************************************************}


procedure premier(var out_put,in_put : chainevar;var identif : chainentier;
                  var out,inn : integer; car1 : char);

{cfr annexe 7.49 }
begin
        pfont(out_put,in_put,identif,out,inn,0,'0');   {<ESC>(0}
        out_put.caract[out] := car1;
        out := out + 1;
        pmvt(out_put,in_put,identif,out,inn,0,1,'A');{<ESC>[1A = <ESC>8}
        pmvt(out_put,in_put,identif,out,inn,0,1,'D');{<ESC>[1D = ^H}
end;

{*************************************************************************}

                {PROCEDURE PMORCEAU}

{*************************************************************************}


procedure pmorceau(var out_put,in_put : chainevar;var identif : chainentier;
                   var out,inn : integer; car2 : char);

{cfr annexe 7.48 }
begin
  out_put.caract[out] := car2;
  out := out + 1;
  pmvt(out_put,in_put,identif,out,inn,0,1,'A');{<ESC>[1A = <ESC>8}
  pmvt(out_put,in_put,identif,out,inn,0,1,'D');{<ESC>[1D = ^H}
end;

{*************************************************************************}
```

```
                {PROCEDURE DERNIER}

{************************************************************************}

procedure dernier(var out_put,in_put : chainevar ;var identif : chainentier;
                  var out,inn : integer; car3 : char);

{cfr annexe 7.48 }
begin
  out_put.caract[out] := car3;
  out := out + 1;
  pfont(out_put,in_put,identif,out,inn,0,'A');   {<ESC>(A}
  pmvt(out_put,in_put,identif,out,inn,0,1,'A');{<ESC>[1A = <ESC>8}
  pmvt(out_put,in_put,identif,out,inn,0,1,'D');{<ESC>[1D = ^H}
end;

{************************************************************************}

                {PROCEDURE ACCOLADE}

{************************************************************************}

procedure accolade(var out_put,in_put : chainevar;var identif : chainentier;
                   var out,inn,compt : integer;
                   car1,car2,car3 : char);


{cfr annexe 7.47 }
var i : integer;

begin
  cde2(out_put,in_put,identif,out,inn,0,'D');{<ESC>D = <ESC>9}
  premier(out_put,in_put,identif,out,inn,car1);
  pmorceau(out_put,in_put,identif,out,inn,'x');
  for i := 2 to ( compt - 1) do pmorceau(out_put,in_put,identif,out,inn,'x');
  pmorceau(out_put,in_put,identif,out,inn,car2); {milieu de la parenthese ou
  for i := (compt + 1) to (2 * compt -1) do
        pmorceau(out_put,in_put,identif,out,inn,'x');
  dernier(out_put,in_put,identif,out,inn,car3);
end;

{************************************************************************}

                {PROCEDURE GDEPARENTH}

{************************************************************************}

procedure gdeparenth(var out_put,in_put : chainevar;
                     var identif : chainentier; var out,inn : integer;
                     var monter,descendre : boolean);

{cfr annexe 7.45 }
{La sequence a modifier est du type
<ESC>9...<ESC>9 !^H<ESC>8<ESC>8...!^H<ESC>8<ESC>8 <ESC>9...<ESC>9
      n-1                          n                        n+1
}

var ttype  : char;
    compt,i: integer;

begin
  ttype := in_put.caract[inn + 1];
  case ttype of
    'a'           :{bold on}
                   pmvt(out_put,in_put,identif,out,inn,4,1,'m');
    'b','f'       :{bold,draft,italic,regular off }
                   pmvt(out_put,in_put,identif,out,inn,4,0,'m');
    'c','d','e':{draft on,italic,helvesan regular off}
                   pmvt(out_put,in_put,identif,out,inn,4,7,'m');
    '(',')','[',']','{','}','|':
        begin
          monter := true;
          descendre := true;
          for i:= 1 to 4 do  ignore(identif,inn,-1);
          compt := 1;
          while in_put.caract[inn] = chr(27) do {<ESC>9}
            begin
              cde2(out_put,in_put,identif,out,inn,2,'D');{<ESC>D = <ESC>9}
              compt := compt + 1;
            end;
          {dans compt on a le nombre de !^H<ESC>8<ESC>8}
          for i := 1 to (6 * compt) do ignore(identif,inn,-1);
```

```
          if compt = 1
            then
               begin
                out_put.caract[out] := ttype;
                out := out + 1;
                for i := 1 to 4 do ignore(identif,inn,-1); {<ESC>9<ESC>9}
               end
            else
                case ttype of
          '(','[': accolade(out_put,in_put,identif,out,inn,compt,'m','x','l');
          ')',']': accolade(out_put,in_put,identif,out,inn,compt,'J','x','k');
              '{': accolade(out_put,in_put,identif,out,inn,compt,'m','u','l');
              '}': accolade(out_put,in_put,identif,out,inn,compt,'J','t','k');
              '|': accolade(out_put,in_put,identif,out,inn,compt,' ','x',' ');
                end;
        end;
        end;
end;

{*************************************************************************}

                  {PROCEDURE TRT_ECRAN}

{*************************************************************************}

procedure trtecran(var in_put,out_put : chainevar ;

                       var monter,descendre,bool_uns : boolean);


{cfr annexe 7.45 }
var inn,out : integer; {position dans le buffer d'entree,de sortie}
    index : integer;  {no de l'action selon le caractere traite}
    identif : chainentier; {utilise pour traitr les <BS>}

begin
  inn :=1;
  out := 1;
  while inn <= in_put.long do
        begin {selection de l'action selon le caractere considere de l'input}
          index := action[ord(in_put.caract[inn]) + 1];
          if (index = 2) and (in_put.caract[inn + 2] <> chr(15))
                then index := 1;
          if (index = 7) and (in_put.caract[inn + 1] <> '|') then index :=1;
          if (index = 8) and ((in_put.caract[inn + 2] <> chr(8))
                or (in_put.caract[inn + 3] <> chr(14)))
                then index := 1; {pas la seq ^O char ^H ^N}


          case index of
           1:copy_maj(out_put,in_put,identif,out,inn,1);
             {caractere imprimable}
           2:pnroff(out_put,in_put,identif,out,inn);
             {2eme jeu de caracteres de nroff - suite du type ^N-ch-^O}
           3:pescape(out_put,in_put,identif,out,inn,monter,descendre,bool_uns)
             {commande "<ESC>..."}
           4:pbackspace(out_put,in_put,identif,out,inn);
             {suite de backspaces a transformer}
           5:copy_maj(out_put,in_put,identif,out,inn,1);
             {caractere recopie a ignorer pour le traitement des <BS>}
           6:signespec(out_put,in_put,identif,out,inn);
             {signe speciaux a transformer}
           7:begin
                racinecarre(out_put,in_put,identif,out,inn);
                monter := true;
                descendre := true;
             end;
             {modification de la racine carree}
           8: gdeparenth(out_put,in_put,identif,out,inn,monter,descendre);
             {modification des parentheses}
           9:begin
                pmvt(out_put,in_put,identif,out,inn,0,1,'C');
                ignore(identif,inn,1);
             end;
             {space -> <ESC>[1C}
        end;
  end;
  out_put.long := out - 1;
end;
```

```
{*************************************************************************}

                {PROCEDURE INITIALISATION}

{*************************************************************************}

procedure initialisation1;
{cfr annexe 7.44 }
var i : integer;

begin

  {VIDEO
   -----}

  for i := 1 to 32 do
   begin
     with video[i] do
        begin
          length := 7;
          car[1] := chr(27);
          car[2] := '(';
          car[3] := 'O';
          car[5] := chr(27);
          car[6] := '(';
          car[7] := 'A';
        end;
   end;

  for i := 33 to 42 do
   begin
     with video[i] do
        begin
          length := 9;
          car[1] := chr(27);
          car[2] := '[';
          car[3] := '7';
          car[4] := 'm';
          car[6] := chr(27);
          car[7] := '[';
          car[8] := 'O';
          car[9] := 'm';
        end;
   end;
   with video[43] do
        begin
          length := 7;
          car[1] := chr(27);
          car[2] := '(';
          car[3] := 'O';
          car[5] := chr(27);
          car[6] := '(';
          car[7] := 'A';
        end;
  for i := 44 to 48 do
   begin
     with video[i] do
        begin
          length := 9;
          car[1] := chr(27);
          car[2] := '[';
          car[3] := '7';
          car[4] := 'm';
          car[6] := chr(27);
          car[7] := '[';
          car[8] := 'O';
          car[9] := 'm';
        end;
   end;
   with video[49] do
        begin
          length := 7;
          car[1] := chr(27);
          car[2] := '(';
          car[3] := 'O';
          car[5] := chr(27);
          car[6] := '(';
          car[7] := 'A';
        end;
  with video[50] do
        begin
```

```
          length := 9;
          car[1] := chr(27);
          car[2] := '[';
          car[3] := '7';
          car[4] := 'm';
          car[6] := chr(27);
          car[7] := '[';
          car[8] := '0';
          car[9] := 'm';
        end;
    for i := 52 to 62 do
     begin
       with video[i] do
          begin
          length := 9;
          car[1] := chr(27);
          car[2] := '[';
          car[3] := '7';
          car[4] := 'm';
          car[6] := chr(27);
          car[7] := '[';
          car[8] := '0';
          car[9] := 'm';
        end;
     end;

    for i := 64 to 79 do
     begin
       with video[i] do
          begin
          length := 7;
          car[1] := chr(27);
          car[2] := '(';
          car[3] := '0';
          car[5] := chr(27);
          car[6] := '(';
          car[7] := 'A';
        end;
     end;
     with video[80] do
          begin
          length := 9;
          car[1] := chr(27);
          car[2] := '[';
          car[3] := '7';
          car[4] := 'm';
          car[6] := chr(27);
          car[7] := '[';
          car[8] := '0';
          car[9] := 'm';
        end;
    for i := 81 to ASCII_COUNT do
     begin
       with video[i] do
          begin
          length := 7;
          car[1] := chr(27);
          car[2] := '(';
          car[3] := '0';
          car[5] := chr(27);
          car[6] := '(';
          car[7] := 'A';
        end;
     end;
    for i := 1 to 32 do                    {INCONNU : 'SPA' --> '?'}
       with video[i] do  car[4] := 'a'; {caractere inconnu}
    for i := 64 to 79 do                   {INCONNU DE '^' --> 'n'}
       with video[i] do car[4] := 'a'; {caractere inconnu}
    for i := 81 to 95 do                   {INCONNU DE 'p' --> '~'}
       with video[i] do car[4] := 'a'; {caractere inconnu}
    with video[33] do car[5] := 'n';
    with video[34] do car[5] := 'a';
    with video[35] do car[5] := 'b';
    with video[36] do car[5] := 'w';
    with video[37] do car[5] := 'd';
    with video[38] do car[5] := 'L';
    with video[39] do car[5] := 'F';
    with video[40] do car[5] := 'G';
    with video[41] do car[5] := 'Y';
    with video[42] do car[5] := 't';
    with video[43] do car[4] := '{';
    with video[44] do car[5] := 'r';
```

```
with video[45] do car[5] := 'l';
with video[46] do car[5] := 'm';
with video[47] do car[5] := 'n';
with video[48] do car[5] := 'I';
with video[49] do car[4] := '{';
with video[50] do car[5] := 'g';
with video[51] do
      begin
        length := 35;
        car[1] := chr(27);      {<ESC>D}
        car[2] := 'D';
        car[3] := chr(27);      {<ESC>(O}
        car[4] := '(';
        car[5] := 'O';
        car[6] := 'o';
        car[7] := chr(27);      {<ESC>(A}
        car[8] := '(';
        car[9] := 'A';
        car[10] := chr(27);     {<ESC>[1A}
        car[11] := '[';
        car[12] := '1';
        car[13] := 'A';
        car[14] := chr(27);     {<ESC>[1D}
        car[15] := '[';
        car[16] := '1';
        car[17] := 'D';
        car[18] := '>';
        car[19] := chr(27);     {<ESC>[1A}
        car[20] := '[';
        car[21] := '1';
        car[22] := 'A';
        car[23] := chr(27);     {<ESC>[1D}
        car[24] := '[';
        car[25] := '1';
        car[26] := 'D';
        car[27] := chr(27);     {<ESC>(O}
        car[28] := '(';
        car[29] := 'O';
        car[30] := 's';
        car[31] := chr(27);     {<ESC>(A}
        car[32] := '(';
        car[33] := 'A';
        car[34] := chr(27);     {<ESC>D}
        car[35] := 'D';
      end;
with video[52] do car[5] := 'e';
with video[53] do car[5] := 'i';
with video[54] do car[5] := 'f';
with video[55] do car[5] := 'p';
with video[56] do car[5] := 'D';
with video[57] do car[5] := 'x';
with video[58] do car[5] := 's';
with video[59] do car[5] := 'Z';
with video[60] do car[5] := 'V';
with video[61] do car[5] := 'u';
with video[62] do car[5] := '@';
with video[63] do
      begin
        length := 37;
        car[1] := chr(27);      {<ESC>[7m}
        car[2] := '[';
        car[3] := '7';
        car[4] := 'm';
        car[5] := chr(27);      {<ESC>D}
        car[6] := 'D';
        car[7] := chr(27);      {<ESC>(O}
        car[8] := '(';
        car[9] := 'O';
        car[10] := 'j';
        car[11] := chr(27);     {<ESC>[1D}
        car[12] := '[';
        car[13] := '1';
        car[14] := 'D';
        car[15] := chr(27);     {<ESC>[1A}
        car[16] := '[';
        car[17] := '1';
        car[18] := 'A';
        car[19] := 'x';
        car[20] := chr(27);     {<ESC>[1D}
        car[21] := '[';
        car[22] := '1';
        car[23] := 'D';
```

```
            car[24] := chr(27);    {<ESC>[1A}
            car[25] := '[';
            car[26] := '1';
            car[27] := 'A';
            car[28] := '1';
            car[29] := chr(27);    {<ESC>(A}
            car[30] := '(';
            car[31] := 'A';
            car[32] := chr(27);    {<ESC>D}
            car[33] := 'D';
            car[34] := chr(27);    {<ESC>[0m}
            car[35] := '[';
            car[36] := '0';
            car[37] := 'm';
          end;

  with video[80] do car[5] := 'z';
end;

procedure init2;

var i : integer;

begin

  {ACTION
   ------}

  action[1] := 5;
  action[2] := 5;
  action[3] := 1;
  action[4] := 5;
  action[5] := 5;
  action[6] := 1;
  action[7] := 1;
  action[8] := 5;
  action[9] := 4;
  action[10] := 5;
  action[11] := 5;
  action[12] := 1;
  action[13] := 5;
  action[14] := 5;
  action[15] := 2;
  action[16] := 8;
  action[17] := 1;
  action[18] := 5;
  action[19] := 1;
  action[20] := 5;
  action[21] := 1;
  action[22] := 1;
  action[23] := 1;
  action[24] := 5;
  action[25] := 1;
  action[26] := 1;
  action[27] := 1;
  action[28] := 3;
  action[29] := 5;
  action[30] := 1;
  action[31] := 1;
  action[32] := 5;
  action[33] := 9;
  for i := 34 to 43 do action[i] := 1;
  action[44] := 6; {+}
  for i := 45 to 47 do action[i] := 1;
  action[48] := 6; {/}
  for i := 49 to 60 do action[i] := 1;
  action[61] := 6; {<}
  action[62] := 1; {=}
  action[63] := 6; {>}
  for i := 64 to 92 do action[i] := 1;
  action[93] := 7; {\}
  for i := 94 to 95 do action[i] := 1;
  action[96] := 1;
  for i := 97 to 126 do action[i] := 1;
  action[127] := 1; {~}
  action[128] := 5;


  {ESCAPE
  --------}
```

```
  for i := 1 to 97 do escape[i] := 0;
  escape[11] := 100;  {LF}
  escape[27] := 101;  {SUB}
  escape[51] := 200;  {2}
  escape[56] := 200;  {7}
  escape[57] := 200;  {8}
  escape[58] := 200;  {9}
  escape[67] := 103;  {B}
  escape[68] := 103;  {C}
  escape[69] := 200;  {D}
  escape[74] := 103;  {I}
  escape[76] := 101;  {K}
  escape[79] := 200;  {N}
  escape[80] := 101;  {O}
  escape[81] := 100;  {P}
  escape[85] := 103;  {T}
  escape[86] := 200;  {U}
  escape[87] := 101;  {V}
  escape[98] := 101;  {a}
  escape[99] := 401;  {b}
  escape[100] := 0;
  escape[101] := 109;  {d}
  escape[102] := 103;  {e}
  escape[103] := 103;  {f}
  escape[104] := 103;  {g}
  escape[105] := 602;  {h}
  escape[106] := 103;  {i}
  escape[107] := 105;  {j}
  escape[108] := 102;  {k}
  escape[109] := 102;  {l}
  escape[110] := 0;
  escape[111] := 101;  {n}
  escape[112] := 102;  {o}
  escape[113] := 110;  {p}
  escape[114] := 110;  {q}
  escape[115] := 101;  {r}
  escape[116] := 502;  {s}
  escape[117] := 100;  {t}
  escape[118] := 301;  {u}
  escape[119] := 702;  {v}
  escape[120] := 108;  {w}
  escape[121] := 101;  {x}
  escape[122] := 100;  {y}
  escape[123] := 0;
  escape[124] := 0;
  escape[125] := 0;
  escape[126] := 0;
  escape[127] := 0;
  escape[128] := 0;

  bool_uns := false;
  nbre_li := 0;
  nbre_li_blan := 0;
  monter := false;
  descendre := false;

end;



{*************************************************************************}
                {PROGRAMME PRINCIPAL}
{*************************************************************************}

{cfr annexe 7.43 }
begin
  initialisation1;
  init2;
  reset(textin);
  rewrite(textout);
  writeln(textout,chr(27),'[?4h');
  while not eof(textin) do
        begin
              {copie dans le tampon d'entree d'1 ligne lue dans textin}
              i := 1;
              while not eoln(textin) do
                 begin
```

```
                read(textin,ch);
                in_put.caract[i] := ch;
                i := i + 1;
              end;
            in_put.long := i - 1;

            nbre_li := nbre_li + 1;
            if in_put.long = 0
              then
                begin
                  nbre_li_blan := nbre_li_blan + 1;
                  out_put.long := 0;
                end
              else
                begin
                  trtecran(in_put,out_put,monter,descendre,bool_uns);
                  if monter = true then
                       begin
                         for i := 1 to (nbre_li_blan + 1) do
                              begin
                               writeln(textout);
                               writeln;
                              end;
                          monter := false;
                       end;
                  if descendre = true then
                      begin
                        monter := true;
                        descendre := false;
                      end;
                  nbre_li_blan := 0;
                end;

            for i := 1 to out_put.long do
              begin
              write(textout,out_put.caract[i]);
              write(out_put.caract[i]);
              end;

            readln(textin);
            writeln(textout);
            writeln;
        end;
  writeln(textout,chr(27),'c');
  writeln(textout);
  writeln(textout);
end.
```